Universidade Federal de São Carlos– UFSCar
Centro de Ciências Exatas e de Tecnologia– CCET
Departamento de Computação– DC
Programa de Pós-Graduação em Ciência da Computação– PPGCC

Aline Marques Del Valle

Aprendizado de Máquina Automatizado Multiobjetivo para Classificação Multirrótulo

São Carlos
2025

Aline Marques Del Valle

Aprendizado de Máquina Automatizado Multiobjetivo para Classificação Multirrótulo

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências Exatas e de Tecnologia da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação.

Área de concentração: Aprendizado de Máquina e Processamento de Língua Natural

Orientador: Ricardo Cerri
Coorientador: Rafael Gomes Mantovani

São Carlos
2025

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Defesa de Tese de Doutorado do candidato Aline Marques Del Valle, realizada em 13/10/2025.

Comissão Julgadora:

Prof. Dr. Ricardo Cerri (UFSCar)

Profa. Dra. Heloisa de Arruda Camargo (UFSCar)

Prof. Dr. Ricardo Bastos Cavalcante Prudêncio (UFPE)

Prof. Dr. Andre Carlos Ponce de Leon Ferreira de Carvalho (USP)

Profa. Dra. Ana Carolina Lorena (UNIFESP)

O Relatório de Defesa assinado pelos membros da Comissão Julgadora encontra-se arquivado junto ao Programa de Pós-Graduação em Ciência da Computação.

Agradecimentos

Agradeço primeiramente a Deus, por estar ao meu lado ao longo desta jornada, me concedendo força, sabedoria e perseverança para concluir mais uma etapa acadêmica. Expresso meus agradecimentos a todos que contribuíram com este trabalho: meu esposo João Guilherme, meus pais Celina e João Carlos, meus irmãos Carla e Tiago, familiares, amigos e colegas.

Sou especialmente grata ao meu orientador Ricardo Cerri e ao meu coorientador Rafael Gomes Mantovani pela orientação, sugestões, ensinamentos e incentivo diante dos desafios do doutorado. O profissionalismo exemplar e a humanidade de ambos são exemplos que gostaria de levar para minha carreira. Espero que continuemos nossa parceria e colaboração.

Agradeço ao IFSULDEMINAS, pela concessão do afastamento que viabilizou a minha qualificação, bem como aos meus amigos e colegas da instituição, cujo apoio foi essencial durante esse período. Agradeço também à UFSCar pela oportunidade de realizar meu doutorado nesta instituição, bem como aos docentes pelo conhecimento compartilhado e formação acadêmica. Por fim, agradeço à Secretaria Geral de Informática da UFSCar pela infraestrutura computacional disponibilizada, essencial para a realização dos experimentos do doutorado.

Resumo

A classificação multirrótulo é um problema desafiador e sua resolução envolve a escolha de um algoritmo de classificação e seus respectivos hiperparâmetros. Contudo, encontrar a melhor combinação algoritmo-hiperparâmetros não é uma tarefa trivial. É nesse contexto que o Aprendizado de Máquina Automatizado (*Automated Machine Learning* - AutoML) surge como solução, automatizando a seleção do melhor algoritmo e sua configuração de hiperparâmetros para problemas de aprendizado de máquina. Esta tese aborda o AutoML e a classificação multirrótulo. Apesar dos recentes avanços na área, principalmente relacionados à construção de classificadores multirrótulo, alguns problemas permanecem em aberto. Neste trabalho, investigamos três desafios nessa área. Investigamos se é viável e eficiente empregar a otimização multiobjetivo em estratégias AutoML para classificação multirrótulo. Para tratar essa questão de pesquisa, propusemos a estratégia EMANUEL (*gEnetic Multi-objective strategy for the Automatic selectioN of mUlti-labEl cLassifiers*), uma estratégia AutoML desenvolvida com o algoritmo NSGA-II com o objetivo de encontrar classificadores que maximizam o Macro F-score enquanto minimizam o tamanho dos modelos. Nossos resultados mostraram que a otimização multiobjetivo pode encontrar uma fronteira de Pareto diversificada, com classificadores que ponderam os diferentes objetivos e que apresentam desempenho competitivo em relação a outros classificadores. Além disso, estudamos técnicas que reduzem o tempo de execução do AutoML sem comprometer os resultados. Para isso, empregamos meta-aprendizado por meio de modelos substitutos para avaliar algoritmos de classificação multirrótulo e estimar os valores dos objetivos da otimização, evitando avaliações diretas durante a execução da estratégia AutoML. A inclusão desses modelos substitutos em novas versões da estratégia EMANUEL manteve a qualidade preditiva dos modelos finais, enquanto reduziu o tempo total de execução do AutoML. Por fim, investigamos o efeito da inclusão da seleção de atributos na estratégia EMANUEL. Para tanto, estendemos o hiperespaço de algoritmos de classificação, adicionando algoritmos de seleção de atributos multirrótulo. As fronteiras de Pareto resultantes da nova versão de EMANUEL continham classificadores que mantive-

ram o desempenho em relação aos objetivos, normalmente utilizando um número menor de atributos.

Palavras-chave: Aprendizado de Máquina Automatizado. Classificação Multirrótulo. Otimização Multiobjetivo. Meta-aprendizado. Seleção de Atributos.

Abstract

Multi-label classification is a challenging problem, and its resolution involves choosing a classification algorithm and its respective hyperparameters. However, finding the best algorithm-hyperparameter combination is not a trivial task. In this context, Automated Machine Learning (AutoML) emerges as a solution, automating the selection of the best algorithm and its hyperparameter configuration for a machine learning problem. For multi-label classification problems, selection occurs in a hyperspace designed for this problem category. This thesis addresses AutoML and multi-label classification. Despite advances in the field, mainly related to constructing multi-label classifiers, some issues remain open. In this work, we investigate three of these areas. We investigate whether employing multi-objective optimization in AutoML strategies for multi-label classification is feasible and efficient. To address this research question, we proposed the EMANUEL strategy (gEnetic Multi-objective strategy for the Automatic selectioN of mUlti-labEl cLassifiers), an AutoML strategy developed with the NSGA-II algorithm to find classifiers that maximize the Macro F-score while minimizing the model size. Our results showed that multi-objective optimization can find a diversified Pareto frontier, with classifiers that weight different objectives and perform competitively with other classifiers. Furthermore, we study techniques that reduce AutoML runtime without compromising results. To this end, we employ meta-learning using surrogate models to evaluate multi-label classifier algorithms and estimate the objective values of optimization, avoiding direct evaluations during AutoML strategy execution. Including these surrogate models in new versions of the EMANUEL strategy maintained the predictive quality of the final models while reducing the total AutoML runtime. Finally, we investigated the effect of including feature selection in the EMANUEL strategy. To this end, we extend the hyperspace of classification algorithms by adding multi-label feature selection algorithms. The Pareto frontiers resulting from the new version of EMANUEL contained classifiers that maintained performance relative to the objectives, typically using fewer features.

Keywords: Automated Machine Learning. Multi-label Classification. Multiobjective Optimization. Meta-Learning. Feature Selection.

Lista de ilustrações

Lista de tabelas

Lista de siglas

AM Aprendizado de Máquina

ARFF *Attribute-Relation File Format*

AutoML *Automated Machine Learning*

AutoMLFS *Automated Multi-label Feature Selection*

AutoMMLC *Automated Multi-objective strategy for Multi-Label Classification*

BCC *Bayesian Classifier Chains*

BNC *Bayesian Network Classifier*

BPNN *Back Propagation Neural Network*

BPMLL *BackPropagation in Multi-Label Learning*

BR *Binary Relevance*

BRq *Binary Relevance – Quick Version*

BRkNNa *BR and KNN in version A*

BRkNNb *BR and KNN in version B*

CASH *Combined Algorithm Selection and Hyperparameter Optimization*

CC *Classifier Chains*

CD *Crowding Distance*

CDN *Conditional Dependency Networks*

CDT *Conditional Dependency Trellis*

CT *Classifier Trellis*

DT *Decision Table*

DTr *Decision Trees*

ECC *Ensembles of Classifier Chains*

EMANUEL *gEnetic Multi-objective strategy for the Automatic selectioN of mUlti-labEl cLassifiers*

EMANUEL$_{FS}$ *gEnetic Multi-objective strategy for the Automatic selectioN of mUlti-labEl cLassifiers and Feature Selection algorithms*

EMANUEL$_{FS+}$ *gEnetic Multi-objective strategy for the Automatic selectioN of mUlti-labEl cLassifiers and Feature Selection algorithms based on surrogate models*

EMANUEL$_{SM}$ *gEnetic Multi-objective strategy for the Automatic selectioN of mUlti-labEl cLassifiers based on Surrogate Models*

FC *Feature Construction*

FE *Feature Extraction*

FEDOT *Framework for Evolutionary Design of Optimal Trees*

FLOP *floating-point operation*

FS *Feature Selection*

FW *Four-Class Pairwise Classification*

HOMER *Hierarchy Of Multi-labEl leaRners*

HTN *Hierarchical Task Network*

IBk *Instance-based algorithm*

IBLR+ *Instance-Based Logistic Regression for Multi-Label Classification*

IGD *Inverted Generational Distance*

IGMF *Information Gain for Multi-label Feather*

IPF *Integrated Preference Functional*

J48 J48

JRip JRip

K* *K Star*

KDIS *Knowledge and Discovery Systems*

kNN *k-Nearest Neighbors*

LIME *Local Interpretable Model-Agnostic Explanations*

LMT *Logistic Model Trees*

LP *Label Powerset*

LR *Logistic Regression*

LRFS *Label Redundancy Feature Selection*

LSMFS *Multi-label Feature Selection considering Label Supplementation*

MCC *Monte-Carlo Classifier Chains*

MDMR *Max-dependency and Min-redundancy*

MLC *Multi-label Classification*

MLkNN *Multi-Label k Nearest Neighbours*

MLP *Multi-Layer Perceptron*

MLSMFS *Multi-label Feature Selection considering Maximum Label Supplementation*

MOGA *Multi-Objective Genetic Algorithm*

MORS *Multi-objective Random Search*

MLTSVM *Multi-Label Support Vector Machine*

MLARAM *Multi-Label Adaptive Resonance Associative Map*

NAS *Neural Architecture Search*

NB *Naïve Bayes*

NSGA-II *Nondominated Sorting Genetic Algorithm II*

OB Otimização Bayesiana

PAES *Pareto Archived Evolution Strategy*

PART *Partial Decision Trees*

PCA *Principal Component Analysis*

PCC *Probabilistic Classifier Chains*

PDP *Partial Dependence Plot*

PMCC *Population of MCC*

PMU *Pairwise Multi-label Utility*

PPT *Pruned Problem Transformation*

PPT-MI *PPT with Mutual Information*

PS *Pruned Sets*

PSt *Pruned Sets with Threshold*

RAkEL *Random K-Label Pruned Sets*

RAkELd *Random K-Label Disjoint Pruned Sets*

RAkELo *Random k-labelsets Overlapping*

REPTree REPTree

RF *Random Forest*

RFE *Recursive Feature Elimination*

RT *Random Tree*

RTh *Ranking and Threshold*

SCLS *Scalable Criterion for Large Label Set*

SGD *Stochastic Gradient Descent*

SL *Simple Logistic*

SMO *Sequential Minimal Optimization*

SPEA *Strength Pareto Evolutionary Algorithm*

SVM *Support Vector Machine*

VP *Voted Perceptron*

XAI *Explainable Artificial Intelligence*

Sumário

Capítulo 1

Introdução

Com a crescente disponibilidade de dados e os avanços tecnológicos, os dados são continuamente analisados e transformados em conhecimento. No Aprendizado de Máquina (AM), esse processo ocorre por meio de diferentes etapas que possibilitam a indução de modelos capazes de representar o conhecimento intrínseco aos dados (MITCHELL, 1997). Esse fluxo de processos, ou sequência de algoritmos que transformam dados de entrada em modelos, é denominado de *pipeline* (ZÖLLER; HUBER, 2021). Entre as tarefas que normalmente compõem um *pipeline* destacam-se a preparação de dados, a engenharia de atributos, a construção e a avaliação dos modelos (HE; ZHAO; CHU, 2021). No entanto, não há um protocolo universal válido para todos os problemas de AM que defina quais tarefas do *pipeline* devem ser adotadas ou quais algoritmos empregados em cada uma delas. Efetivamente, os *pipelines* são configurados e parametrizados a partir da experimentação para problemas específicos.

Assim, a abordagem tradicional para projetar soluções de AM é essencialmente manual e exaustiva. A partir dos dados de um problema, especialistas realizam experimentações, selecionando algoritmos e calibrando seus hiperparâmetros para as diferentes etapas do *pipeline* em busca dos melhores modelos. Essa abordagem é: repetitiva e trabalhosa; requer conhecimento sobre os algoritmos de cada etapa do *pipeline*; é difícil de ser replicada devido à subjetividade inerente às escolhas realizadas; e pode levar a soluções subótimas, já que é inviável testar exaustivamente as combinações de algoritmos e hiperparâmetros de forma manual (FEURER; HUTTER, 2019).

A automatização desse processo poderia produzir soluções de AM mais eficientes e reduzir os erros humanos. Dessa forma, emergiu o Aprendizado de Máquina Automatizado (do inglês *Automated Machine Learning* (AutoML)), um campo de pesquisa que visa automatizar a construção de modelos de AM (ZÖLLER; HUBER, 2021). Segundo Brazdil

et al. (2022), o AutoML automatiza o projeto e o treinamento dos modelos de AM, explorando combinações de algoritmos e de configurações de hiperparâmetros com base em um hiperespaço com todas as decisões de projeto – incluindo a arquitetura de redes neurais ou a estrutura de *pipelines* do AM.

Feurer, Springenberg e Hutter (2015) também definem o AutoML como um problema que combina a seleção de algoritmos e a otimização de hiperparâmetros (do inglês *Combined Algorithm Selection and Hyperparameter Optimization* (CASH)), modelando-o como um problema de otimização. Nessa perspectiva, dado um conjunto de dados e um hiperespaço composto por algoritmos de AM e seus respectivos hiperparâmetros, o objetivo é encontrar uma solução ótima que maximize o desempenho preditivo no conjunto de dados de treinamento.

Assim, um algoritmo de otimização pode ser utilizado para conduzir a busca por uma solução ótima (mono-objetivo), ou um conjunto de soluções ótimas (multiobjetivo), com base em uma ou mais medidas de avaliação pré-definidas. Durante essa busca, diferentes combinações de algoritmos e configurações de hiperparâmetros são avaliadas até que uma condição de parada pré-definida seja satisfeita. No entanto, a complexidade computacional da otimização pode ser influenciada por diversos fatores, como a natureza do problema (mono ou multi-alvo, por exemplo), a dimensionalidade dos conjuntos de dados, a quantidade e a complexidade dos algoritmos e configurações de hiperparâmetros avaliados durante a otimização.

Uma alternativa recente a esse desafio no AutoML é o uso do Meta-aprendizado (BRAZDIL et al., 2022). Segundo Vanschoren (2019), o Meta-aprendizado caracteriza-se pela capacidade de aprender a partir de experiências anteriores em tarefas que já foram realizadas. Sua aplicação envolve a coleta de metadados descritivos das tarefas anteriores, o treinamento e o teste de um metamodelo. No contexto do AutoML, esses metamodelos podem ter diferentes objetivos, como recomendar algoritmos e configurações de hiperparâmetros promissoras, prever o desempenho ou o tempo de execução de algoritmos específicos (BRAZDIL et al., 2022), ou mesmo estimar medidas de complexidade dos dados (GARCIA et al., 2020).

Embora o AutoML tenha sido aplicado principalmente para resolver problemas tradicionais de classificação e de regressão, sua utilização em cenários de classificação multirrótulo permanece pouco explorada (SÁ; FREITAS; PAPPA, 2018; WEVER et al., 2019). Na classificação monorrótulo, as instâncias estão associadas a um único rótulo. Já na classificação multirrótulo, as instâncias podem estar associadas a um conjunto de rótulos binários (TSOUMAKAS; KATAKIS; VLAHAVAS, 2009). A natureza multirrótulo representa problemas do mundo real de forma mais realista, mas introduz maior complexidade aos algoritmos de AM e às metodologias de avaliação da classificação multirrótulo (CHERMAN, 2014).

Na literatura[1], as estratégias AutoML desenvolvidas especificamente para classificação multirrótulo[2] (SÁ; PAPPA; FREITAS, 2017; SÁ; FREITAS; PAPPA, 2018; WEVER et al., 2019; SÁ et al., 2020; WEVER et al., 2021) automatizam a tarefa de construção dos modelos do *pipeline* de AM, utilizando algoritmos de otimização mono-objetivo. Essas estratégias adotam uma representação hierárquica do hiperespaço, incluindo algoritmos de classificação monorrótulo e multirrótulo e o espaço de hiperparâmetros desses algoritmos. Para a exploração do hiperespaço, as estratégias utilizam diferentes algoritmos de otimização, como os Algoritmos Genéticos, busca em Rede de Tarefas Hierárquicas, Hiperbanda e Otimização Bayesiana, por exemplo. Durante a otimização, os modelos são induzidos e avaliados por meio de uma medida que serve como critério de desempenho durante a busca pela solução ótima.

Apesar dos avanços recentes da aplicação de AutoML para classificação multirrótulo, diversas lacunas de pesquisa permanecem em aberto. As estratégias existentes concentram-se na automatização da construção dos modelos de AM, mas as demais tarefas do *pipeline* não são tratadas (SÁ; PAPPA; FREITAS, 2017; SÁ; FREITAS; PAPPA, 2018). Outra limitação está no uso exclusivo da otimização mono-objetivo, ignorando a natureza multiobjetivo de muitos problemas reais (WEVER et al., 2019). O único trabalho que aborda AutoML e otimização multiobjetivo para classificação multirrótulo é um trabalho preliminar decorrente desta tese (VALLE; MANTOVANI; CERRI, 2023a). Além disso, observa-se uma ausência de mecanismos que tornem o AutoML mais eficiente, como a incorporação de técnicas de meta-aprendizado, por exemplo (WEVER et al., 2019; WEVER et al., 2021).

Muitas dessas lacunas também têm aplicabilidade no cenário monorrótulo. No entanto, é importante salientar que, no contexto multirrótulo, os problemas tornam-se consideravelmente mais complexos. Isso ocorre porque cada instância pode estar associada simultaneamente a múltiplos rótulos, o que gera um espaço de saída maior do que na classificação monorrótulo. Além disso, os rótulos podem apresentar dependências que afetam o desempenho dos classificadores, exigindo modelos capazes de capturar essas relações. Esse cenário também tende a intensificar problemas como o desbalanceamento de rótulos, aumentando a dificuldade de aprendizado e de avaliação. Dessa forma, abordar as lacunas citadas representa um desafio significativo no desenvolvimento de estratégias AutoML para classificação multirrótulo.

Para abordar essas lacunas, propomos *gEnetic Multi-objective strategy for the Automatic selectioN of mUlti-labEl cLassifiers* (EMANUEL), uma estratégia baseada na otimização multiobjetivo para classificação multirrótulo. Além disso, desenvolvemos versões

[1] O escopo desta pesquisa não abrange a automatização da busca por arquiteturas neurais (do inglês *Neural Architecture Search* (NAS)). Por esse motivo, os trabalhos relacionados a essa abordagem não foram incluídos no texto.

[2] Também existem ferramentas AutoML, como `auto-sklearn`, que foram originalmente projetadas para problemas convencionais do AM, mas que incluíram suporte à classificação multirrótulo. No entanto, essas ferramentas apresentam limitações para o cenário multirrótulo.

de EMANUEL, que automatizam a seleção de atributos e empregam modelos substitutos como técnica de meta-aprendizagem para acelerar a execução do AutoML. De maneira geral, os resultados mostraram que a otimização multiobjetivo pode encontrar um conjunto de soluções com resultados competitivos em relação aos objetivos. Tanto a versão de EMANUEL com seleção de atributos quanto a versão sem a seleção apresentaram resultados consistentes. De forma similar, as versões com e sem modelos substitutos obtiveram desempenhos comparáveis. Porém, o tempo de execução das estratégias que incorporaram os modelos substitutos foi significativamente menor. Na próxima seção, detalhamos os problemas de pesquisa, as questões investigadas e as hipóteses propostas.

1.1 Problemas, Questões e Hipóteses de Pesquisa

As estratégias convencionais de AutoML para classificação multirrótulo adotam tipicamente a otimização mono-objetivo, supondo que uma única medida pode avaliar o desempenho geral de um classificador. Durante a execução dessas estratégias, a otimização mono-objetivo avalia iterativamente diferentes classificadores, selecionando aquele que otimiza a medida-alvo (objetivo). No entanto, os classificadores multirrótulo podem ser avaliados por diferentes métricas. Assim, o classificador selecionado pela estratégia AutoML certamente terá bons resultados para a medida-alvo, mas pode ter desempenho pior para as demais medidas.

Diante dessa limitação, temos a primeira questão de pesquisa: ***É viável e eficaz empregar otimização multiobjetivo em estratégias AutoML para classificação multirrótulo?*** A otimização multiobjetivo é uma alternativa promissora à otimização mono-objetivo, pois ela é capaz de encontrar um conjunto de soluções não dominadas (fronteira de Pareto) que otimiza diferentes medidas de desempenho (objetivos) simultaneamente (DEB; DEB, 2014). Os usuários têm flexibilidade para escolher nesse conjunto de soluções aquelas que melhor atendam às suas expectativas. Neste trabalho, adotamos objetivos que ponderam desempenho e eficiência computacional, maximizando o Macro F-score e minimizando o tamanho do modelo, respectivamente. Dessa forma, formulamos a seguinte hipótese de pesquisa:

Hipótese 1: Estratégias AutoML multiobjetivo para classificação multirrótulo geram um conjunto diversificado de soluções não dominadas, proporcionando maior flexibilidade para a seleção de classificadores que mantêm desempenho competitivo, comparável a outros algoritmos.

Durante o processo de otimização, múltiplos algoritmos de classificação multirrótulo com diferentes configurações de hiperparâmetros são treinados e avaliados iterativamente, até que uma condição de parada seja satisfeita. Esse processo apresenta custo computacional significativo, principalmente quando utiliza-se algoritmos de classificação complexos

ou a dimensionalidade do conjunto de dados é alta. Diante desse desafio, surge nossa segunda questão de pesquisa: ***Como reduzir o tempo de execução do AutoML sem comprometer a qualidade dos modelos?*** Uma alternativa para esse problema é o uso de técnicas de meta-aprendizado, mais especificamente dos modelos substitutos. Esses modelos podem ser utilizados para estimar o desempenho dos classificadores multirrótulo durante o processo de otimização, eliminando a necessidade de avaliações completas e onerosas. Dessa forma, temos como hipótese de pesquisa:

Hipótese 2: Estratégias AutoML para classificação multirrótulo que empregam modelos substitutos para prever o desempenho de um classificador são capazes de manter a qualidade preditiva dos modelos finais enquanto reduzem o tempo total de execução do AutoML.

O último problema abordado na tese reside no fato de que as estratégias AutoML para classificação multirrótulo concentram-se na automatização da construção dos modelos de AM e as demais tarefas do *pipeline* do AM não são automatizadas. Isso nos leva à nossa terceira questão de pesquisa: ***É possível automatizar a engenharia de atributos, em particular a seleção de atributos, sem comprometer a qualidade preditiva das soluções AutoML?*** Segundo Kashef, Nezamabadi-pour e Nikpour (2018), a seleção de atributos diminui a dimensionalidade dos dados e pode melhorar o desempenho da classificação multirrótulo. Dessa forma, a proposta consiste em uma estratégia AutoML capaz de automatizar tanto a seleção de atributos quanto a construção dos modelos preditivos. Nessa estratégia, o algoritmo de otimização multiobjetivo avalia de forma iterativa diferentes combinações de seletores de atributos e de classificadores multirrótulo a fim de encontrar um conjunto de soluções não dominadas (fronteira de Pareto) que otimizem as medidas pré-estabelecidas. Com base nessa fundamentação, nossa hipótese de pesquisa é:

Hipótese 3: Estratégias AutoML que incorporam a seleção automatizada de atributos são capazes de encontrar soluções que mantenham ou até melhorem o desempenho preditivo em relação a outros algoritmos/estratégias que não incorporam essa funcionalidade.

1.2 Objetivos

O objetivo geral desta tese é investigar questões em aberto na área do AutoML, propor e avaliar experimentalmente novas estratégias AutoML para classificação multirrótulo.

1.2.1 Objetivos Específicos

- ❑ Verificar se a otimização multiobjetivo pode contribuir para o AutoML, gerando um conjunto de soluções não dominadas, que flexibilizam a seleção de classificadores que mantém o desempenho preditivo.

- ❏ Investigar o potencial do uso de meta-aprendizado no AutoML para acelerar a busca pelos melhores modelos de AM, mantendo o desempenho preditivo dos classificadores multirrótulo.
- ❏ Averiguar se a automatização de tarefas relacionadas à seleção de atributos pode contribuir com a busca por modelos que utilizam um subconjunto de atributos, mas que mantém o desempenho preditivo.

1.3 Contribuições

A tese aborda diferentes aspectos do AutoML para classificação multirrótulo e traz as seguintes contribuições para essa área de pesquisa:

- ❏ Proposta de uma nova estratégia AutoML que incorpora critérios multiobjetivo para a busca dos melhores modelos de AM.
- ❏ Redução do tempo de execução da estratégia AutoML por meio do uso de modelos substitutos que prevêem o desempenho de um algoritmo de classificação multirrótulo, dada uma configuração de hiperparâmetros.
- ❏ Modelagem de um hiperespaço com algoritmos de seleção de atributos multirrótulo.
- ❏ Proposta de uma nova estratégia AutoML multiobjetivo que automatiza a seleção de atributos e a construção modelos de classificação multirrótulo.
- ❏ Condução de uma prova de conceito a fim de averiguar a redução do tempo de execução da estratégia AutoML multiobjetivo que automatiza a seleção de atributos e a construção modelos de classificação multirrótulo usando modelos substitutos.
- ❏ Reprodutibilidade dos experimentos: códigos e dados estão disponíveis publicamente para que os experimentos possam ser replicados.

1.4 Organização da tese

O restante desta tese está organizado da seguinte forma:

- ❏ O Capítulo 2 apresenta o referencial teórico com os temas importantes da tese: classificação multirrótulo, AutoML e otimização multiobjetivo (NSGA-II).
- ❏ O Capítulo 3 revisa os trabalhos relacionados ao tema desta pesquisa: AutoML para classificação multirrótulo.
- ❏ O Capítulo 4 descreve a metodologia experimental geral, utilizada em todos os experimentos.

- ❏ O Capítulo 5 detalha a proposta AutoML multiobjetivo para a classificação multirrótulo.
- ❏ O Capítulo 6 investiga o uso dos modelos substitutos para redução do tempo de execução da estratégia proposta.
- ❏ O Capítulo 7 apresenta um experimento de escopo reduzido sobre a automatização da seleção de atributos.
- ❏ O Capítulo 8 redefine a estratégia AutoML, incluindo a automatização da seleção de atributos.
- ❏ O Capítulo 9 é uma prova de conceito, onde investiga-se o uso de modelos substitutos na estratégia AutoML que inclui a automatização da seleção de atributos.
- ❏ O Capítulo 10 apresenta as conclusões da tese, discute as limitações e as propostas de trabalhos futuros.

Para complementar os capítulos da tese, incluímos os seguintes apêndices:

- ❏ O Apêndice A define o hiperespaço multirrótulo, identificando algoritmos de classificação monorrótulo e multirrótulo, bem como o espaço de hiperparâmetros desses algoritmos.
- ❏ O Apêndice B sumariza os resultados de EMANUEL e dos algoritmos *baselines*.
- ❏ O Apêndice C descreve os experimentos relacionados aos modelos substitutos.
- ❏ O Apêndice D sumariza os resultados da versão de EMANUEL com modelos substitutos e dos algoritmos *baselines*.
- ❏ O Apêndice E apresenta uma breve descrição de alguns algoritmos de seleção de atributos.
- ❏ O Apêndice F define o hiperespaço de algoritmos de seleção de atributos.

Capítulo 2

Referencial Teórico

Este capítulo apresenta uma revisão dos conceitos fundamentais que embasam esta tese. A Seção 2.1 fundamenta a classificação multirrótulo. A Seção 2.2 detalha a estrutura de *pipeline* do AutoML e as técnicas que contribuem para o seu funcionamento. A Seção 2.3 explora conceitos e algoritmos da otimização multiobjetivo. Por fim, a Seção 2.4 apresenta as considerações finais do capítulo.

2.1 Classificação Multirrótulo

Na classificação multirrótulo (do inglês *Multi-label Classification* (MLC)), cada instância do conjunto de dados é rotulada com um conjunto de rótulos/classes binários (MADJAROV et al., 2012; XU et al., 2020). Um conjunto de dados contendo n instâncias é representado na forma $\{(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \ldots (\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}$. $\mathbf{X}$ é o conjunto de instâncias contendo os atributos de entrada, onde $\mathbf{x}^{(i)}$ é um vetor com os d atributos da instância i $(x_1^{(i)}, x_2^{(i)}, \ldots x_d^{(i)})$. $\mathbf{Y}$ é o conjunto de rótulos de saída, onde $\mathbf{y}^{(i)}$ é um vetor com os q rótulos da instância i $(y_1^{(i)}, y_2^{(i)}, \ldots, y_q^{(i)})$. Cada rótulo j da instância i $(y_j^{(i)})$ pode assumir um valor binário indicando se o rótulo é relevante ou não para a instância. O objetivo da classificação multirrótulo é aprender uma função $f : \mathbf{X} \rightarrow \mathbf{Y}$ que encontra os rótulos das instâncias, dado o conjunto de dados de entrada.

Segundo Tsoumakas, Katakis e Vlahavas (2009), problemas de classificação multirrótulo podem ser manipulados de duas formas distintas: transformação de problema (abordagem independente de algoritmos) e adaptação de algoritmo (abordagem dependente de algoritmos). A primeira abordagem converte o problema multirrótulo em múltiplos problemas monorrótulo e utiliza algoritmos monorrótulo tradicionais. A segunda abordagem cria algoritmos que tratam diretamente o problema multirrótulo, lidando com todas as

classes simultaneamente. Normalmente, esses algoritmos estendem os algoritmos monorrótulo existentes.

Uma definição similar é apresentada por Madjarov et al. (2012), que categoriza os algoritmos de aprendizado multirrótulo em três abordagens: transformação de problema; adaptação de algoritmo; e algoritmos *ensemble*. As duas primeiras abordagens são equivalentes às propostas por Tsoumakas, Katakis e Vlahavas (2009). A diferença fica com a nova categoria de algoritmos, baseados em comitês (*ensembles*), compostos por um conjunto de classificadores multirrótulo como método base, visando melhorar a capacidade de generalização do algoritmo e reduzir o *overfitting* (MOYANO et al., 2018).

Na abordagem de transformação de problema, o algoritmo de classificação multirrótulo *Binary Relevance* (BR) é um dos mais simples e conhecidos. Esse algoritmo decompõe o conjunto de rótulos em q rótulos para a classificação binária, onde q é o número total de rótulos do problema original. Para cada rótulo, um novo conjunto de dados é gerado e treinado individualmente com um classificador monorrótulo. As predições resultantes são então combinadas para obter a classificação multirrótulo (TSOUMAKAS; KATAKIS; VLAHAVAS, 2009). Desse modo, constrói-se um classificador binário distinto para cada rótulo, que determina se o rótulo está ou não associado a uma instância.

Outro algoritmo importante e relevante nessa categoria é o algoritmo *Classifier Chains* (CC) (READ et al., 2009). CC é um algoritmo que também trabalha com q classificadores binários, porém, à medida que as classificações são realizadas, seus resultados passam a compor os atributos de entrada do conjunto de dados como novos campos. O objetivo é criar uma dependência entre rótulos de forma que as próximas classificações possam ser aprimoradas (CHERMAN, 2014). O algoritmo CC é sensível à ordem com que os rótulos são selecionados para a classificação.

Na abordagem de adaptação de algoritmo, os algoritmos lidam diretamente com a classificação, que acontece em um único passo e sem transformações no conjunto de dados. Normalmente, estes algoritmos são baseados em algoritmos monorrótulos conhecidos. Por exemplo, *Multi-Label k Nearest Neighbours* (MLkNN) é uma adaptação do algoritmo *k-Nearest Neighbors* (kNN) (ZHANG; ZHOU, 2005) e *BackPropagation in Multi-Label Learning* (BPMLL) é uma adaptação do algoritmo *Multi-Layer Perceptron* (MLP) (ZHANG; ZHOU, 2006).

Já os algoritmos *Ensembles of Classifier Chains* (ECC) (READ et al., 2009) e *Random K-Label Pruned Sets* (RAkEL) (TSOUMAKAS; VLAHAVAS, 2007) são exemplos de algoritmos da abordagem baseada em *ensembles*. ECC treina um conjunto de classificadores CC utilizando diferentes subconjuntos de dados de treinamento e diferentes ordens de encadeamento dos classificadores binários (CHERMAN, 2014). A classificação de um novo exemplo é realizada por votação, a partir das classificações obtidas do conjunto de classificadores. Por outro lado, o algoritmo RAkEL trabalha com um pequeno subconjunto de rótulos (*k-labelsets*), onde k representa o número de rótulos em cada subconjunto. Assim,

esse algoritmo constrói iterativamente um conjunto de classificadores *Label Powerset* (LP) para cada combinação aleatória de rótulos.

Independentemente da abordagem dos algoritmos de classificação multirrótulo, eles geram predições que são avaliadas por um mesmo tipo de medidas de avaliação. Essas medidas verificam se o rótulo atribuído a uma instância está correto. No entanto, a avaliação da classificação multirrótulo é diferente da classificação monorrótulo, uma vez que a classificação de uma instância pode estar parcialmente correta, contendo rótulos corretos e rótulos incorretos simultaneamente. Na classificação multirrótulo, o critério de avaliação pode ser baseado em instâncias ou baseado em rótulos. Em ambos os critérios, a classificação obtida é avaliada para cada instância/rótulo individualmente e depois as avaliações são combinadas por meio de média. Nesta tese, revisamos apenas as medidas de avaliação baseadas em rótulo (Seção 2.1.1), devido à relevância dessas medidas para os experimentos. Para mais informações sobre as medidas baseadas em instâncias, consulte Zhang e Zhou (2014).

2.1.1 Medidas de Avaliação Baseadas em Rótulos

Na classificação binária monorrótulo, as classes podem ser positivas ou negativas e os classificadores podem ser avaliados por medidas de avaliação binárias, baseadas no número de verdadeiros positivos (VP), verdadeiros negativos (VN), falsos positivos (FP) e falsos negativos (FN). As medidas de Acurácia, Precisão, Revocação e F-score são exemplos dessas medidas que podem ser calculadas a partir das predições retornadas pelos algoritmos. As Equações 1, 2, 3 e 4 definem formalmente essas medidas. A notação $\uparrow$ nas equações indica que valores maiores correspondem aos melhores resultados para cada medida.

$$\uparrow \text{Acurácia} = \frac{VP + VN}{VP + VN + FP + FN} \tag{1}$$

$$\uparrow \text{Precisão} = \frac{VP}{VP + FP} \tag{2}$$

$$\uparrow \text{Revocação} = \frac{VP}{VP + FN} \tag{3}$$

$$\uparrow F\_score = \frac{2VP}{2VP + FP + FN} \tag{4}$$

Na classificação multirrótulo, utilizam-se as medidas de avaliação binária definidas para problemas monorrótulo (Equações 1 a 4), em conjunto com operações de média. Essas operações são conhecidas como *micro-averaging* e *macro-averaging* (TSOUMAKAS; KATAKIS; VLAHAVAS, 2009). Dado $B(VP, VN, FP, FN)$, tal que B é uma medida de

avaliação binária e os valores VP, VN, FP e FN obtidos da matriz de confusão, as versões *micro-averaging* (B_{Micro}) e *macro-averaging* (B_{Macro}) da medida B para classificação multirrótulo são definidas pelas Equações 5 e 6 (PEREIRA et al., 2018).

$$\uparrow B_{Macro} = \frac{1}{q}\sum_{i=1}^{q} B(VP_i, VN_i, FP_i, FN_i) \tag{5}$$

$$\uparrow B_{Micro} = B(\sum_{i=1}^{q} VP_i, \sum_{i=1}^{q} VN_i, \sum_{i=1}^{q} FP_i, \sum_{i=1}^{q} FN_i) \tag{6}$$

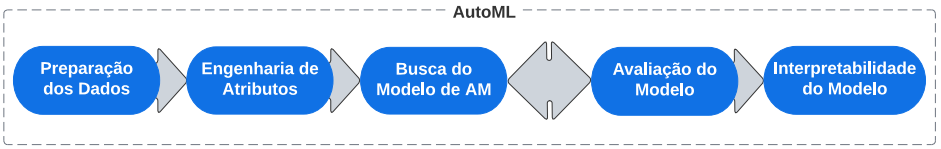
Na medida B_{Macro}, a medida de avaliação B é calculada individualmente para cada rótulo e, em seguida, uma média desses valores é computada. Já na medida B_{Micro}, os valores de VP, VN, FP e FN são totalizados considerando todos os rótulos e, a partir dessa soma, calcula-se a medida binária B. Esse procedimento equivale a calcular a medida binária B para cada instância e, posteriormente, calcular a média.

2.2 AutoML

O *Automated Machine Learning* (AutoML) propõe melhorar a maneira atual de construir aplicações de AM, tornando-as mais automatizadas. De acordo com Chen, Song e Hu (2021), o AutoML busca soluções de AM de alto desempenho utilizando esforço manual reduzido e um orçamento computacional limitado. A automatização poderia tornar as soluções de AM mais populares, pois elas seriam encontradas automaticamente, sendo mais acessíveis para usuários inexperientes e, consequentemente, mais aplicáveis a novos domínios.

Esta definição é complementada por Zöller e Huber (2021). Segundo os autores, o AutoML consiste na automação de *pipelines* de AM. Dado um conjunto de dados, o papel do AutoML é encontrar uma sequência de tarefas do *pipeline* que otimize o desempenho no conjunto de dados em questão. Essas tarefas incluem desde o pré-processamento de dados até a interpretabilidade dos modelos de AM. Para cada tarefa, há diferentes algoritmos e hiperparâmetros, e o AutoML deve selecionar os que minimizam as perdas e obtêm os melhores resultados.

Figura 1 – Exemplo de *pipeline* genérico.



Fonte: Adaptado de He, Zhao e Chu (2021).

A Figura 1 ilustra um exemplo de um *pipeline* genérico de estrutura fixa para tarefas de predição, composto pelas tarefas de preparação de dados, engenharia de atributos, busca do modelo de AM, avaliação e interpretabilidade do modelo. Com base neste *pipeline*, a Seção 2.2.1 apresenta as tarefas do *pipeline* de AM com mais detalhes e o funcionamento do AutoML, enquanto a Seção 2.2.2 descreve algumas técnicas que contribuem com o AutoML, seja reduzindo o custo computacional e/ou melhorando a qualidade dos modelos encontrados.

2.2.1 Tarefas do *Pipeline* de AM

Nesta seção, são descritas as tarefas que compõem o *pipeline* apresentado na Figura 1, com ênfase no AutoML.

2.2.1.1 Preparação de Dados

Os conjuntos de dados são essenciais para o desenvolvimento de soluções de AM (PAULLADA et al., 2021). Atualmente, os dados são armazenados em larga escala por empresas e instituições e podem ser facilmente recuperados. Eles também estão disponíveis em repositórios públicos[1]. No entanto, na ausência de dados adequados, é possível obtê-los por meio de simulações ou mesmo da internet, seja pela coleta manual ou por dispositivos automatizados.

Intuitivamente, acredita-se que conjuntos de dados maiores podem facilitar o treinamento de modelos de AM, pois possuem mais instâncias para a indução dos modelos e, consequentemente, um potencial maior para generalização. No entanto, a qualidade dos dados é uma preocupação, pois pode resultar em modelos incorretos e não confiáveis. Exemplos comuns de má qualidade incluem dados desbalanceados, inconsistentes ou com valores faltantes. Segundo Chu et al. (2016), antes de treinar os modelos, é essencial realizar a limpeza dos dados para tratar problemas como dados faltantes, erros de digitação, duplicados, que violam regras de negócio, *outliers*, entre outros. Para He, Zhao e Chu (2021), a limpeza de dados requer conhecimento humano especializado sobre o domínio do problema e pode ser automatizada mediante a definição prévia das operações de limpeza ou a transformação dessa tarefa em um problema de otimização de hiperparâmetros.

Além disso, os dados disponíveis no conjunto podem ser insuficientes para a construção de modelos. Nesse caso, novos dados podem ser gerados a partir dos dados originais para auxiliar o treinamento. Esse processo é denominado *data augmentation* (ROH; HEO; WHANG, 2021). Por exemplo, novas imagens podem ser obtidas por meio de rotações, alterações de brilho e recortes, assim como dados textuais podem ser enriquecidos com sinônimos e traduções. Segundo He, Zhao e Chu (2021), a etapa de *data augmenta-*

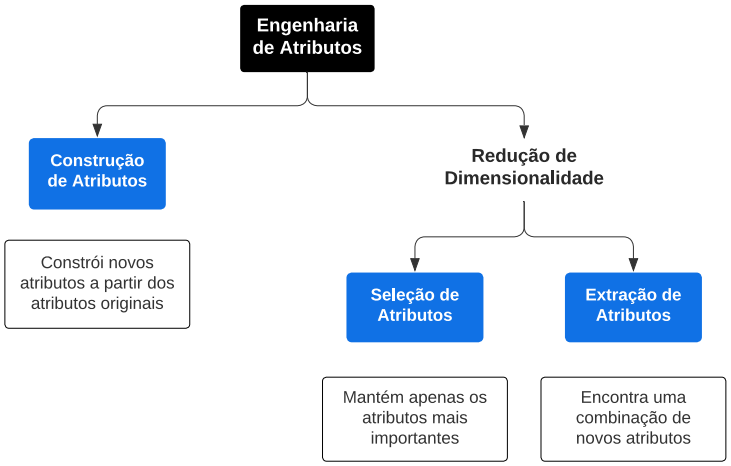
[1] `Kaggle`, `UCI`, `OpenML`, `KDIS` e `Mulan` são alguns exemplos desses repositórios.

tion também pode ser automatizada, mas as operações de aumento de dados devem ser selecionadas por humanos.

2.2.1.2 Engenharia de Atributos

A engenharia de atributos visa transformar os dados brutos, melhorando os atributos existentes (KARMAKER et al., 2021). No AutoML essa melhoria deve ocorrer de forma automática. Essa etapa é essencial para todo o *pipeline*, pois o desempenho do modelo depende diretamente dos atributos disponíveis. Wang et al. (2022) subdividem a engenharia de atributos em construção, extração e seleção de atributos, visando maximizar a qualidade dos atributos obtidos dos dados brutos. A Figura 2 ilustra um esquema dessa subdivisão conforme a descrição de Wang et al. (2022).

Figura 2 – Subdivisão da Engenharia de Atributos.



- **Construção de Atributos (do inglês *Feature Construction* (FC))**: é uma tarefa que expande o conjunto original de atributos, estabelecendo relações até então desconhecidas entre eles, na tentativa de melhorar a generalização dos modelos. Padronização, normalização e discretização são métodos de transformação tradicionalmente utilizados para a construção de novos atributos no pré-processamento de dados. Além disso, outros operadores podem ser empregados na construção de atributos (ZÖLLER; HUBER, 2021). Operadores unários, como o logaritmo, podem ser aplicados diretamente a um atributo, criando um novo atributo. De forma similar, novos atributos são criados a partir de operadores binários como $+$, $-$, $*$, $/$ e a correlação. No caso da correlação, por exemplo, na área da saúde peso e altura podem originar o índice de massa corporal, que tende a ser mais informativo do que os atributos originais. Já os operadores de alta ordem funcionam de modo análogo ao

comando *group by* do SQL, agregando atributos por meio de funções como mínimo, máximo, soma e média.

- **Extração de Atributos (do inglês *Feature Extraction* (FE))**: é uma tarefa cujo objetivo é reduzir a dimensionalidade dos dados mediante a aplicação de funções específicas de mapeamento. Essa técnica é utilizada quando há redundância de atributos ou a sua dimensionalidade é muito alta. Esse processo organiza um conjunto de dados em um novo conjunto com menor dimensionalidade. Os novos atributos, embora distintos, preservam as informações relevantes contidas nos atributos originais. A técnica *Principal Component Analysis* (PCA)[2] é um dos algoritmos de extração de atributos mais populares.

- **Seleção de atributos (do inglês *Feature Selection* (FS))**: elimina dados irrelevantes, preservando apenas atributos que tenham uma relação significativa com os rótulos e que contribuam para a predição (WANG et al., 2022). Nesse processo, um subconjunto de atributos é selecionado dentre os atributos originais para tornar o treinamento mais eficiente e melhorar o desempenho dos modelos de AM, removendo dados redundantes ou incorretos (ZÖLLER; HUBER, 2021).

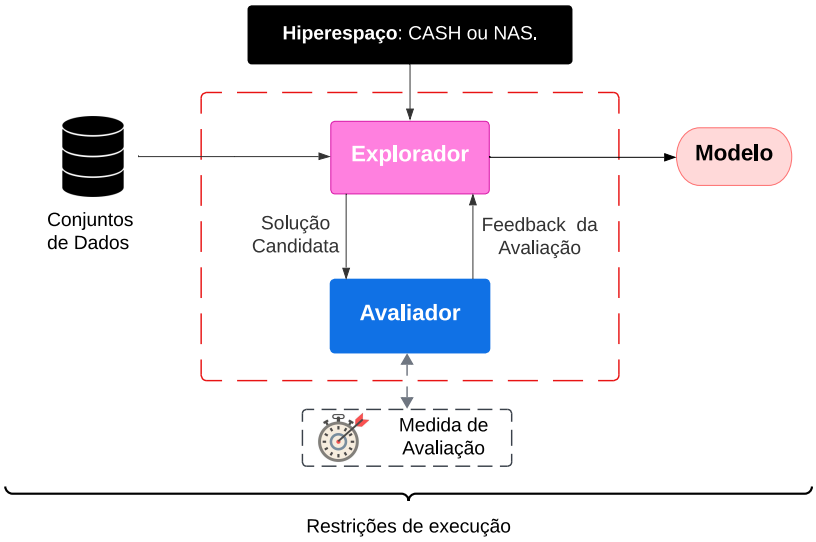
Em síntese, a construção de atributos expande o conjunto de dados, enquanto a extração e a seleção de atributos reduzem a dimensionalidade dos dados. No entanto, a extração de atributos gera novos atributos que compactam os originais e a seleção de atributos filtra atributos dentre os existentes. Espera-se que os atributos resultantes da engenharia de atributos melhorem a representação do problema, permitindo o treinamento de modelos de alto desempenho (KARMAKER et al., 2021).

2.2.1.3 Busca do Modelo de AM

Para a construção dos modelos de AM, o objetivo do AutoML é encontrar automaticamente o algoritmo, bem como os valores dos hiperparâmetros (abordagem CASH) ou a arquitetura de rede (abordagem NAS) que resultem no modelo com o melhor desempenho preditivo. Para isso, é necessário definir o hiperespaço, o algoritmo de otimização para a exploração desse hiperespaço, e a medida de avaliação para guiar a busca. Dessa forma, um avaliador e um explorador, representados por uma medida de avaliação e um algoritmo de otimização, respectivamente, assumem o lugar do humano no AutoML (CHEN; SONG; HU, 2021), examinando e avaliando sistematicamente soluções candidatas em busca de uma solução ótima. A Figura 3 ilustra esse processo de avaliação e de exploração para a automatização da busca pelo modelo de AM com base no hiperespaço.

[2] A PCA é um algoritmo simples que lida com dados de alta dimensão, representando-os de forma mais compacta. O algoritmo PCA analisa os componentes principais para derivar novas variáveis, que são uma combinação linear das variáveis originais (WEBB, 2011).

Figura 3 – Funcionamento iterativo de estratégias AutoML.



Fonte: Adaptado de Chen, Song e Hu (2021).

Quando o AutoML é modelado como problemas CASH, o hiperespaço é formado por um conjunto de algoritmos de predição e seus respectivos espaços de hiperparâmetros. Para exemplificar, algoritmos como *Naïve Bayes* (NB) e kNN podem compor um hiperespaço para problemas de classificação monorrótulo. Nesse exemplo, um dos hiperparâmetros incluídos no hiperespaço, juntamente com seus possíveis valores, é o número de vizinhos (k) do algoritmo kNN. Consequentemente, se o algoritmo kNN for selecionado durante a execução do AutoML, torna-se necessário atribuir um valor para o hiperparâmetro k com base no hiperespaço definido.

Por outro lado, para problemas NAS, o hiperespaço define os paradigmas estruturais que compõem as arquiteturas neurais, que podem ser selecionados pelos algoritmos de otimização. Para Elsken et al. (2019), os hiperespaços NAS podem ser categorizados em estruturados em cadeia, baseados em células, ou hierárquicos. O hiperespaço estruturado em cadeia é parametrizado pelo número de camadas, tipo de operação que cada camada executa (agrupamento, convolução, *pooling*, *skip connections*, dentre outras) e hiperparâmetros associados às operações. A arquitetura resultante é uma cadeia sequencial de operações composta por N camadas, onde a camada L recebe sua entrada da camada $L-1$ e a sua saída é a entrada para a camada $L+1$.

O hiperespaço baseado em células é composto por blocos que normalmente se repetem nas estruturas neurais, chamados células. As células são microestruturas formadas por nós que recebem entradas anteriores e aplicam um conjunto de operações – como agrupamento, convolução, *pooling* e *skip connections* – para gerar representações mais informativas. A arquitetura resultante é construída a partir da repetição e combinação dessas células.

Por fim, o hiperespaço hierárquico, introduzido por Liu et al. (2018), baseia-se na criação de *motifs* (subpadrão formado por operações) organizados em diferentes níveis. No primeiro nível estão as operações primitivas. No segundo nível, os *motifs* conectam essas operações primitivas. No terceiro nível, os *motifs* de nível superior conectam os *motifs* do segundo nível. Dessa forma, um padrão de nível superior é representado como um grafo direcionado acíclico composto por subpadrões[3].

Uma vez que o hiperespaço é definido, a busca automatizada pelo modelo de AM pode ocorrer. De acordo com a Figura 3, dado um conjunto de dados, o explorador obtém do hiperespaço as possíveis soluções candidatas. Para problemas CASH, uma solução candidata é representada por um algoritmo de AM e pelos valores dos hiperparâmetros desse algoritmo. A partir das soluções candidatas, os modelos são construídos e avaliados pelo avaliador utilizando uma medida de avaliação pré-estabelecida. O avaliador retorna um *feedback* da sua avaliação para o explorador, que atualiza as soluções candidatas para que novas avaliações sejam feitas. O processo de exploração e avaliação ocorre de forma iterativa até que uma condição de parada seja satisfeita. O resultado da tarefa de busca do modelo de AM é o melhor modelo dentre os modelos avaliados.

O explorador é representado pelo algoritmo de otimização. Portanto, a estratégia de busca do explorador depende do algoritmo de otimização empregado. Há diferentes algoritmos de otimização que podem ser empregados no AutoML, desde métodos simples como a Busca em Grade (*Grid Search*) e a Busca Aleatória (*Random Search*) até algoritmos mais elaborados como a Otimização Bayesiana (*Bayesian Optimization*) e o Aprendizado por Reforço (*Reinforcement Learning*). Neste trabalho, não serão revisados os algoritmos de otimização tradicionalmente aplicados no AutoML, pois um dos objetivos da pesquisa é empregar a otimização multiobjetivo. Dessa forma, revisamos os algoritmos de otimização multiobjetivo empregados nos experimentos na Seção 2.3.

Embora a Figura 3 ilustre o funcionamento do AutoML para indução de um modelo de AM, o funcionamento é o mesmo para automatização das demais tarefas do *pipeline*. Assim, uma solução candidata pode incluir algoritmos para preparação de dados, engenharia de atributos e do modelo do AM. Uma abordagem possível consiste em utilizar hiperespaços específicos para cada tarefa do *pipeline*.

2.2.1.4 Avaliação do Modelo

Durante a busca automatizada dos modelos de AM, diferentes soluções candidatas são geradas, e por consequência, é necessário medir o desempenho dos modelos induzidos com tais soluções. A técnica de avaliação mais simples é a avaliação direta, que consiste no treinamento e na avaliação do modelo. Esta abordagem é muito precisa, porém custosa para ser aplicada de forma repetida.

[3] Mais informações sobre o hiperespaço NAS podem ser encontradas em Elsken et al. (2019) e He, Zhao e Chu (2021).

De acordo com Zöller e Huber (2021), sistemas AutoML têm um tempo de resposta alto e o processo de otimização-avaliação é lento, dependendo do conjunto de dados. Uma alternativa é trabalhar com subconjuntos de dados de treinamento (*sub-sampling*) de forma a diminuir o volume de dados de treino e acelerar o processo de avaliação. Além disso, a avaliação das soluções candidatas também pode ser interrompida antecipadamente (*early stop*). Na parada antecipada, o treinamento de um modelo de AM é iterativo e converge para um erro mínimo. Ao final de cada iteração, a curva de aprendizagem é avaliada. Caso não haja melhora na avaliação ou se o tempo de convergência torne-se elevado, por exemplo, a avaliação pode ser interrompida.

Outra alternativa para agilizar o treinamento e as avaliações do modelo é a reutilização de hiperparâmetros de treinamentos anteriores e o uso de avaliadores substitutos (*surrogate evaluator*) como técnica de avaliação (YAO et al., 2018). A primeira explora as configurações de hiperparâmetros promissoras de iterações ou execuções anteriores, mapeando valores de hiperparâmetros e seus desempenhos. Funciona como uma memória das configurações já avaliadas. Por outro lado, avaliadores substitutos utilizam um modelo de AM para predizer o desempenho de determinadas configurações, com base nas experiências de avaliações anteriores. As técnicas listadas nesta seção podem ser aplicadas individualmente ou em conjunto.

2.2.1.5 Interpretabilidade do Modelo

Normalmente, os modelos de AM são construídos e avaliados a partir do seu desempenho, por medidas como acurácia e precisão, por exemplo. No entanto, não há informações sobre a lógica interna e sobre o funcionamento desses modelos. Assim, não é possível entendê-los e, consequentemente, compreender como as decisões são tomadas. Dessa forma, a interpretabilidade é essencial para garantir a correção dos modelos quando os problemas envolvem a tomada de decisão, ajudando a entender as decisões tomadas e tornando os modelos mais confiáveis (CARVALHO; PEREIRA; CARDOSO, 2019).

O campo de pesquisa que se concentra na interpretabilidade no AM é a Inteligência Artificial Explicável, do inglês *Explainable Artificial Intelligence* (XAI) (LENT; FISHER; MANCUSO, 2004). Para Miller (2019), a "interpretabilidade é o grau que o humano pode entender a causa de uma decisão"[4] e, para Murdoch et al. (2019), a interpretabilidade é a "extração de conhecimento relevante de um modelo de AM sobre as relações contidas nos dados ou aprendidas pelo modelo"[5]. Assim, quanto mais fácil for entender as decisões do modelo, mais interpretável ele é.

Segundo Molnar (2022), a interpretabilidade pode ser intrínseca ao modelo ou *post hoc*. A interpretabilidade intrínseca refere-se aos modelos de AM naturalmente interpre-

[4] Tradução livre do original: "interpretability is the degree to which a human can understand the cause of a decision".

[5] Tradução livre do original: "extraction of relevant knowledge from a machine-learning model concerning relationships either contained in data or learned by the model".

táveis. Exemplos desses modelos são os construídos a partir de algoritmos como regressão linear, regressão logística e árvore de decisão. A interpretabilidade *post hoc* utiliza métodos que analisam os modelos após o treinamento. Esses métodos podem ser classificados como (MOLNAR, 2022):

- **Específicos do modelo ou agnóstico ao modelo:** métodos específicos são projetados para determinados tipos de algoritmos, como redes neurais, e exploram suas particularidades internas. Já os métodos agnósticos ao modelo podem ser aplicados a qualquer algoritmo de AM, oferecendo maior flexibilidade por não dependerem da estrutura interna do modelo.

- **Local ou global:** os métodos locais explicam predições individuais e tentam entender como o modelo chegou a determinada predição. Já os métodos globais tentam entender o comportamento do modelo como um todo, considerando todas as predições.

Partial Dependence Plot (PDP) é um exemplo de método agnóstico global (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). O PDP mostra a interação entre um ou dois atributos do conjunto de dados e o resultado da predição do modelo, a fim de explicar a influência dos atributos na predição. Por outro lado, o *Local Interpretable Model-Agnostic Explanations* (LIME) é um exemplo de um método agnóstico local (RIBEIRO; SINGH; GUESTRIN, 2016). Esse método tenta explicar o comportamento de uma predição para uma instância do conjunto de dados, criando um modelo substituto interpretável a partir de instâncias próximas à instância de interesse[6].

A interpretabilidade do modelo pode ser automatizada juntamente com as demais tarefas do *pipeline* do AM (MOLNAR, 2022). No entanto, a interpretabilidade normalmente é aplicada *post-hoc* ao AutoML, por meio de métodos específicos, a fim de aumentar a transparência dos modelos encontrados (HASAN et al., 2025; HOOF; VANSCHOREN, 2021). Além disso, pesquisas recentes começaram a incluir a interpretabilidade como um dos critérios de otimização (KARL et al., 2023; SCHNEIDER; BISCHL; THOMAS, 2023).

2.2.2 Técnicas que Contribuem com o AutoML

Aliada às técnicas de otimização e avaliação, técnicas avançadas como Meta-aprendizado (do inglês *Meta-learning*) e Transferência de Aprendizado (do inglês *Transfer Learning*) podem ser empregadas para contribuir com o AutoML. As Seções 2.2.2.1 e 2.2.2.2 descrevem Meta-aprendizado e Transferência de Aprendizado, respectivamente.

[6] Mais informações sobre métodos de interpretabilidade de modelos podem ser encontradas em Molnar (2022).

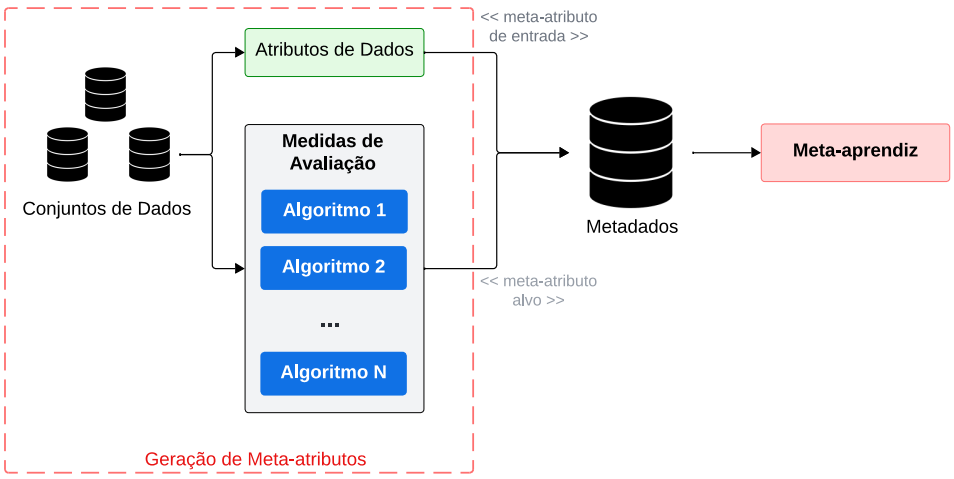
2.2.2.1 Meta-aprendizado

Dado um conjunto de dados, diferentes soluções candidatas são avaliadas no AutoML. No entanto, o que é feito com todas as informações geradas nessas execuções? Conhecemos as medidas de desempenho (acurácia, precisão, dentre outras) e de complexidade dos algoritmos (tempo de execução, tamanho dos modelos, entre outros), bem como os algoritmos e suas configurações de hiperparâmetros mais promissoras para um determinado problema. Todas essas informações constituem experiências anteriores sobre um determinado problema.

O meta-aprendizado (ou aprender a aprender) é uma área do AM que busca aprender com as experiências obtidas por meio da observação sistemática do desempenho de diversos algoritmos de aprendizado em múltiplas tarefas similares (VANSCHOREN, 2019). As experiências acumuladas geram meta-conhecimento, que serve como base para melhorar futuros processos de aprendizagem. Esse meta-conhecimento abrange informações extraídas de tarefas previamente exploradas, dos algoritmos utilizados e dos modelos gerados (BRAZDIL et al., 2022).

A Figura 4 ilustra a estrutura geral do meta-aprendizado. Inicialmente, são extraídos os meta-atributos dos conjuntos de dados e dos algoritmos de aprendizado (as propriedades estatísticas dos conjuntos de dados e os hiperparâmetros dos algoritmos, por exemplo). Esses atributos correspondem aos meta-atributos de entrada. Também são coletados atributos relacionados ao desempenho desses algoritmos, denominados meta-atributos alvo. Os meta-atributos de entrada e alvo constituem os metadados ou meta-conhecimento, utilizados para a indução de um modelo de AM, denominado meta-aprendiz.

Figura 4 – Estrutura geral do meta-aprendizado.



Fonte: Adaptado de Mantovani (2018) e Brazdil et al. (2022).

Os metadados e os meta-aprendizes podem ser construídos para diferentes propósitos. Segundo Yao et al. (2018) e Zöller e Huber (2021), para problemas CASH e NAS, o meta-aprendiz pode ser empregado para refinar o hiperespaço, avaliando a importância dos hiperparâmetros e mantendo apenas os mais importantes ou identificando as regiões mais promissoras do hiperespaço. O meta-aprendiz também pode ser treinado como avaliadores substitutos para avaliar algoritmos e suas configurações de hiperparâmetros, seja para classificar as mais promissoras ou mesmo para prever o desempenho das soluções em relação a uma medida de avaliação (acurácia e tempo de execução, por exemplo).

Na literatura, há diversos trabalhos que utilizam o meta-aprendizado no AutoML. Normalmente, ele é empregado para recomendar configurações de hiperparâmetros (ZHONGGUO et al., 2017; LORENA et al., 2018), para recomendar os valores iniciais dos hiperparâmetros (GOMES et al., 2012; FEURER et al., 2015), para prever se o ajuste de hiperparâmetros é necessário (RIDD; GIRAUD-CARRIER, 2014; MANTOVANI et al., 2019) e para prever o desempenho para uma determinada configuração de hiperparâmetros (REIF et al., 2014; EGGENSPERGER et al., 2018; CARNEIRO et al., 2023). Nesta tese, utilizamos o meta-aprendizado para prever os valores dos objetivos de uma solução candidata.

2.2.2.2 Transferência de Aprendizado

A maioria dos algoritmos de AM assume que os dados de treinamento e de teste compartilham o mesmo espaço de atributos e a mesma distribuição estatística. Quando ocorre uma mudança na distribuição, esses algoritmos precisam ser reconstruídos com novos dados. Essa abordagem, porém, implica retrabalho significativo, além de demandar nova coleta de dados e processo de treinamento. A *Transferência de Aprendizado* surge como uma solução para esse desafio, permitindo o uso do conhecimento adquirido em problemas anteriores para resolver novas tarefas de forma mais eficiente, resultando em soluções de melhor qualidade, com menor esforço.

Segundo Pan e Yang (2010), a Transferência de Aprendizado tem como objetivo aplicar o conhecimento de uma ou mais tarefas de origem em uma tarefa de destino. Dessa forma, novos modelos de AM são derivados de modelos pré-treinados, herdando sua estrutura fundamental e configurações de hiperparâmetros. De acordo com Vanschoren (2019), dado um modelo treinado para a tarefa de origem T_s, ele pode ser transferido para uma nova tarefa de destino T_t, de forma que T_s e T_t sejam semelhantes ou que haja uma conexão entre as tarefas.

No contexto das redes neurais, a arquitetura do modelo de origem pode ser utilizada como inicialização para o modelo de destino, no qual um modelo pré-treinado é transferido e refinado a partir dos dados de treinamento da tarefa de destino. Antes da transferência, pode haver necessidade de adaptação do modelo de origem (VANSCHOREN, 2019) e, após a transferência, a maioria das camadas pode ser compartilhada sem alterações, e

apenas as camadas finais requerem ajuste fino para obter a rede de destino (ZHUANG et al., 2020). No entanto, algumas abordagens de transferência de aprendizado utilizam a mesma arquitetura para as tarefas de origem e destino.

No AutoML, o NAS realiza a busca exploratória de arquiteturas neurais, consumindo muitos dados e recursos computacionais devido à necessidade de avaliar um grande número de arquiteturas (LIAN et al., 2019). Nesse contexto, a transferência de aprendizado surge como estratégia para otimizar o NAS, permitindo a transferência de arquiteturas de rede (incluindo hiperparâmetros e parâmetros obtidos no treinamento) de tarefas de origem para tarefas de destino, reduzindo assim a carga computacional inerente ao processo de busca (WEN; PENG; TING, 2021). Outra aplicação relevante em NAS é a transferência de arquiteturas de rede com preservação de função (YAO et al., 2018). Nesse caso, o hiperespaço é explorado em busca de novas arquiteturas que mantêm o comportamento funcional da arquitetura de origem. Isso permite encontrar redes que preservem a mesma capacidade de representação do modelo original, sem comprometer o desempenho inicial, porém arquiteturas mais eficientes.

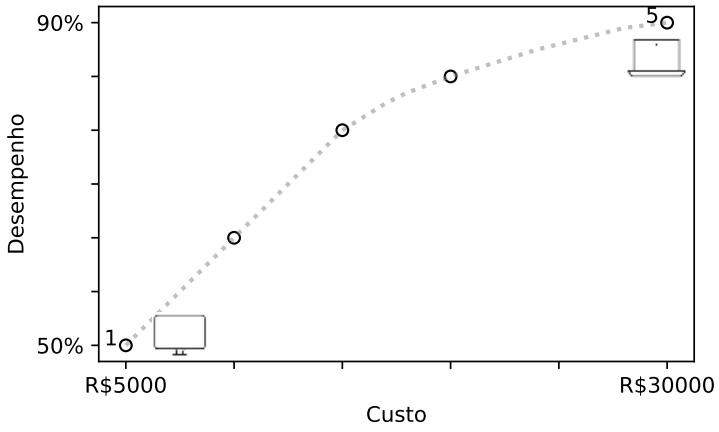
2.3 Otimização Multiobjetivo

Muitos problemas práticos de AM têm como objetivo otimizar apenas uma métrica, como a acurácia ou o tempo de treinamento/teste dos modelos, por exemplo. No entanto, existem casos em que o objetivo é otimizar duas ou mais variáveis simultaneamente. Esses problemas são conhecidos como problemas multiobjetivo.

Um problema de otimização multiobjetivo é definido como a busca por valores mínimos ou máximos de duas ou mais funções objetivo, sujeitas a restrições de igualdade e desigualdade. Já os problemas de otimização mono-objetivo buscam otimizar uma única função objetivo e as variáveis da função também podem estar sujeitas a restrições. Ao contrário da otimização mono-objetivo, na otimização multiobjetivo deseja-se encontrar um conjunto de soluções ótimas que otimizam mais de uma função objetivo ao mesmo tempo. As soluções desse conjunto possuem valores diferentes para os objetivos, nas quais uma solução apresenta ganho em relação a um objetivo, mas provavelmente apresenta perdas em relação aos demais objetivos (DEB; DEB, 2014).

Para exemplificar, considere a tomada de decisão envolvida na compra de um computador, com custos variando de R$2.000,00 (computador 1) até R$30.000,00 (computador 5), conforme mostra a Figura 5. Se o custo do computador fosse o único objetivo a ser maximizado, a resposta na otimização mono-objetivo seria adquirir o computador mais caro. No entanto, na prática, os consumidores avaliam múltiplos fatores, buscando idealmente maximizar o desempenho enquanto minimizam o custo. Um consumidor com maior poder aquisitivo provavelmente compraria o computador mais caro, mas ainda haveria várias opções de computadores com diferentes custos e desempenhos à venda. A

Figura 5 – Exemplo da relação de Custo x Desempenho na compra de computadores.



decisão ótima varia conforme o perfil do consumidor, que deve escolher o computador com melhor relação custo-desempenho, considerando o seu orçamento.

Muitos problemas multiobjetivos já foram resolvidos transformando os múltiplos objetivos em um único objetivo e utilizando a otimização mono-objetivo (DEB; DEB, 2014). No entanto, a abordagem multiobjetivo modela diretamente tais problemas de forma mais prática. Este trabalho utiliza a otimização multiobjetivo na estratégia AutoML proposta. Assim, as Seções 2.3.1 e 2.3.2 revisam, respectivamente, os conceitos de dominância e fronteira de Pareto. A Seção 2.3.3 descreve os algoritmos multiobjetivos utilizados nos experimentos e a Seção 2.3.4 apresenta metodologias de avaliação da fronteira de Pareto.

2.3.1 Dominância

A maioria dos algoritmos multiobjetivo utiliza o conceito de dominância para encontrar o conjunto de soluções ótimas. Para explicar este conceito, as notações $\triangleleft$ e $\triangleright$ serão utilizadas entre duas soluções i e j para indicar que i é melhor ($i \triangleleft j$) ou pior ($i \triangleright j$) que j para um objetivo específico. Uma solução $x^{(1)}$ domina a solução $x^{(2)}$ se e somente se (DEB; DEB, 2014; KARL et al., 2023):

- $x^{(1)}$ não é pior que $x^{(2)}$ para todos os objetivos, ou $f_i(x^{(1)}) \not\triangleright f_i(x^{(2)})$ para todos $i = \{1, 2, \ldots, n\}$.
- $x^{(1)}$ é melhor que $x^{(2)}$ em pelo menos um dos objetivos, ou $f_i(x^{(1)}) \triangleleft f_i(x^{(2)})$ em pelo menos um $i \in \{1, 2, \ldots, n\}$.

Ou seja, $x^{(1)}$ domina $x^{(2)}$ se os valores dos objetivos de $x^{(1)}$ são equivalentes ou melhores que os valores dos objetivos de $x^{(2)}$ e $x^{(1)}$ supera $x^{(2)}$ em pelo menos um dos objetivos. A Figura 6 exemplifica o conceito de dominância para um problema de minimização com dois objetivos. Cada ponto no gráfico representa os valores dos objetivos de uma solução.

A solução A é a solução a ser avaliada. Ela domina as soluções D e E, pois os valores dos objetivos de A são menores ou equivalentes aos objetivos de D e E. Em específico, A supera D em ambos os objetivos e supera E no Objetivo 1. A solução A não domina as soluções B e C, pois as regras definidas acima não são satisfeitas.

Figura 6 – Exemplo de soluções dominadas e não dominadas.

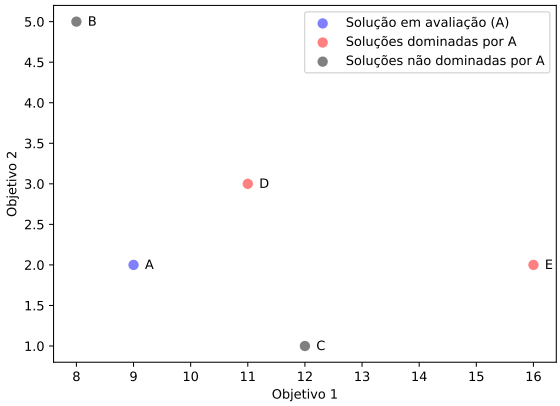
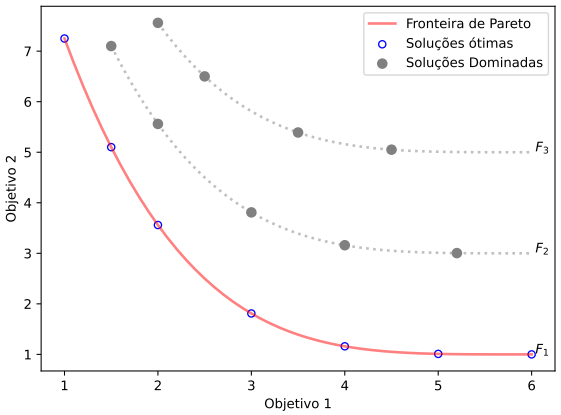


Figura 7 – Exemplo da Fronteira de Pareto.



Fonte: Adaptado de Verma, Pant e Snasel (2021).

2.3.2 Otimalidade de Pareto

A partir do conceito de dominância, é possível comparar todos os pares de soluções em relação aos objetivos para identificar as soluções dominadas e as soluções não dominadas. O conjunto de soluções não dominadas por nenhuma outra solução é formado pelas soluções que mais otimizam as funções objetivos e são o resultado da otimização multiobjetivo (DEB; DEB, 2014; SHARMA; CHAHAR, 2022). Esse conjunto de soluções denomina-se soluções ótimas de Pareto ou fronteira de Pareto.

A Figura 7 ilustra a fronteira de Pareto para um problema de minimização com dois objetivos. Cada ponto no gráfico representa a avaliação dos objetivos de uma solução. As soluções dominadas correspondem aos pontos na cor preta e as soluções ótimas de Pareto (não dominadas) são representadas na cor azul. As soluções dominadas têm avaliação pior a pelo menos uma das soluções ótimas de Pareto. A fronteira de Pareto é representada na cor vermelha e é a primeira fronteira não dominada (F_1).

Se as soluções da primeira fronteira não dominada forem eliminadas do conjunto de soluções, uma segunda fronteira não dominada pode ser identificada (F_2). Da mesma forma, as soluções da segunda fronteira não dominada também podem ser eliminadas, permitindo encontrar a terceira fronteira não dominada (F_3). Assim, diversas fronteiras de Pareto podem ser identificadas no espaço dos objetivos ($F = \{F_1, F_2, \dots\}$) e as soluções podem ter diferentes *ranks* de dominação. Na Figura 7, a linha em vermelho representa a

primeira fronteira ($rank = 1$) e as linhas tracejadas representam a segunda ($rank = 2$) e a terceira ($rank = 3$) fronteiras, respectivamente. Muitos algoritmos de otimização multiobjetivo baseiam-se no conceito da otimalidade de Pareto, podendo explorar os diferentes *ranks* de dominação.

2.3.3 Algoritmos de Otimização Multiobjetivo

Segundo Sharma e Chahar (2022), há diferentes algoritmos baseados em dominância, entre eles: *Pareto Archived Evolution Strategy* (PAES), *Strength Pareto Evolutionary Algorithm* (SPEA), *Multi-Objective Genetic Algorithm* (MOGA), *Multi-objective Random Search* (MORS) e *Nondominated Sorting Genetic Algorithm II* (NSGA-II). Neste trabalho, estamos interessados nos algoritmos MORS e NSGA-II, descritos nas Seções 2.3.3.1 e 2.3.3.2, respectivamente.

2.3.3.1 *Multi-objective Random Search* (MORS)

A busca aleatória (*Random Search*) é um dos algoritmos de otimização mono-objetivo mais populares. Dado o número T de soluções candidatas a serem avaliadas, o algoritmo avalia as T soluções candidatas geradas aleatoriamente em busca da melhor solução (FEURER; HUTTER, 2019). De acordo com Karl et al. (2023), a versão multiobjetivo da busca aleatória pode ser derivada do algoritmo de otimização mono-objetivo e do conceito de dominância. Para isso, MORS seleciona T soluções candidatas de forma aleatória, avalia essas soluções com as N funções objetivo e retorna o conjunto de soluções não dominadas, ou seja, a fronteira de Pareto.

Algoritmo 1 MORS

Require: T
  Selecione uma solução inicial S_0;
  Avalie S_0 com as funções objetivo;
  Inicialize a fronteira de Pareto com S_0: $fronteira = \{S_0\}$
  while $T \geq 1$ **do**
    Selecione uma solução S;
    Avalie S com as funções objetivo;
    Encontre as soluções não dominadas, comparando os valores dos objetivos de S com os valores dos objetivos das soluções em $fronteira$;
    Atualize $fronteira$ com as soluções não dominadas;
    $T = T - 1$
  end while
  return $fronteira$

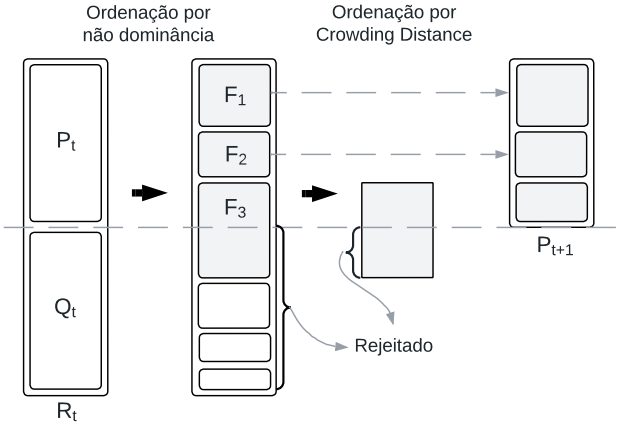
Feng, Shen e Xu (2016) e Babor et al. (2022) descrevem a técnica de MORS de forma similar. No entanto, à medida que as soluções candidatas são avaliadas, as soluções dominadas e não dominadas são identificadas. Assim, uma fronteira de Pareto atualizada

é mantida durante toda a execução de MORS e a dominância de uma solução candidata é avaliada apenas com as soluções da fronteira de Pareto. O Algoritmo 1 apresenta o pseudocódigo do algoritmo MORS. Em linhas gerais, esse algoritmo avalia T soluções candidatas em relação aos objetivos, cujos valores são utilizados para atualizar a fronteira de Pareto a cada iteração.

2.3.3.2 *Nondominated Sorting Genetic Algorithm II* (NSGA-II)

NSGA-II é um algoritmo evolucionário proposto por Deb et al. (2002), que trabalha com uma população de indivíduos que evolui durante as gerações. A Figura 8 será utilizada para explicar o algoritmo NSGA-II na *t-th* geração. Dada a população P_t de tamanho N, os operadores binários de seleção por torneio[7], *crossover* e mutação são utilizados para criar uma nova população de descendentes Q_t, também de tamanho N. As populações P_t e Q_t são combinadas na população R_t de tamanho $2N$. A população R_t é classificada de acordo com a não dominação e os indivíduos das primeiras fronteiras de $F = \{F_1, F_2, \dots\}$ vão sendo incluídos à nova população P_{t+1}, limitados ao tamanho da população N. Na Figura 8, os indivíduos das fronteiras de Pareto F_1 e F_2 foram incluídos à nova população P_{t+1}, mas a inclusão dos indivíduos da fronteira de Pareto F_3 extrapola o tamanho da população.

Figura 8 – Esquema do funcionamento do algoritmo NSGA-II.



Fonte: Deb et al. (2002).

Uma vez que os indivíduos de uma fronteira não podem ser incluídos à população P_{t+1}, apenas os indivíduos com maior *Crowding Distance* (CD) são incluídos na nova população. CD é um indicador de densidade que mede a densidade na vizinhança de uma solução (DEB; DEB, 2014). Quanto maior o valor do CD de uma solução, mais isolada

[7] A seleção por torneio sorteia k indivíduos de forma aleatória, compara a aptidão dos indivíduos e seleciona o indivíduo mais apto (FANG; LI, 2010). Na seleção binária por torneio $k = 2$.

é a solução e também mais importante para manter a dispersão das soluções na fronteira de Pareto. Com a nova população P_{t+1} de tamanho N, uma nova geração se inicia. O funcionamento do algoritmo NSGA-II segue conforme descrito acima até que um critério de parada seja alcançado.

O cálculo da aptidão na primeira geração difere daquele realizado nas gerações seguintes. Na primeira geração, uma população inicial P_0 de tamanho N é criada de forma aleatória. Cada indivíduo de P_0 possui uma aptidão equivalente ao *rank* de não dominação. Indivíduos da primeira fronteira de Pareto possuem aptidão 1, da segunda fronteira de Pareto aptidão 2, e assim sucessivamente. Diversas fronteiras de Pareto podem ser encontradas em P_0 e todos os indivíduos de P_0 possuem uma aptidão correspondente à fronteira de Pareto à qual pertencem, onde indivíduos com aptidão menor são considerados indivíduos melhores.

Nas demais gerações, a aptidão de um indivíduo passa a considerar também o valor de CD. O indivíduo mais apto é aquele com melhor *rank* de dominância. E, em caso de empate no *rank* de não dominação, aquele com maior CD. Assim, para selecionar indivíduos para o *crossover* e para a mutação, tanto o *rank* de não dominação quanto o CD são considerados a partir da segunda geração.

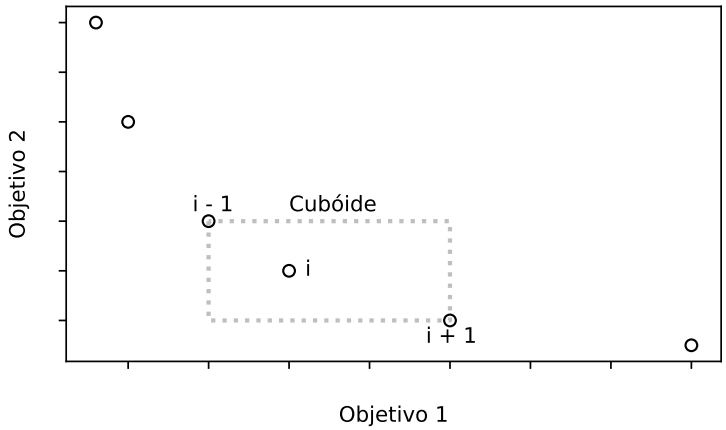
Crowding Distance

Crowding Distance (CD) é uma estimativa de densidade, utilizada no algoritmo NSGA-II para encontrar uma fronteira de Pareto ótima, onde as soluções têm uma boa dispersão pela fronteira. A Figura 9 mostra uma fronteira de Pareto para um problema de minimização com dois objetivos. Na figura, é destacada a solução i, bem como suas soluções vizinhas $i-1$ e $i+1$. Uma região densa terá vizinhos imediatos próximos e uma região menos densa terá vizinhos imediatos mais distantes. CD busca quantificar o quão distantes estão as soluções na fronteira e o algoritmo NSGA-II prioriza as soluções mais distantes.

Dada a fronteira de Pareto, os indivíduos são ordenados de acordo com os objetivos. O CD de uma solução i é calculado pela distância média das soluções vizinhas de i para cada um dos objetivos (DEB et al., 2002; DEB; DEB, 2014; KARL et al., 2023). Na Figura 9, as distâncias das soluções vizinhas à solução i para cada objetivo correspondem aos lados do retângulo. Nesse caso, CD é o comprimento médio do perímetro do cuboide.

O CD de uma solução i (cd^i) pode ser expresso pelas Equações 7 e 8 (DEB et al., 2002; DEB; DEB, 2014). Ele é calculado pela estimativa de densidade de cada objetivo (cd^i_n). A estimativa de densidade de um objetivo n é a diferença entre os valores do objetivo n das soluções vizinhas $i-1$ e $i+1$, em relação à diferença entre os valores do objetivo n das soluções dos extremos da fronteira de Pareto (f^0_n e f^L_n) à qual pertence a solução i. As soluções do extremo da fronteira de Pareto não possuem vizinhos imediatos (antecessor ou sucessor) e, assim, não podem ter o CD calculado pela Equação 8. Essas soluções

Figura 9 – Cálculo do *Crowding Distance*.



Fonte: Deb et al. (2002).

possuem os melhores valores para cada objetivo individualmente e assumem os melhores valores para o CD (∞).

$$cd^i = \sum_{n=1}^{N} cd_n^i \tag{7}$$

$$cd_n^i = \frac{f_n^{i+1} - f_n^{i-1}}{f_n^L - f_n^0} \tag{8}$$

2.3.4 Avaliação da Fronteira de Pareto

Avaliar a qualidade de uma solução na otimização mono-objetivo é uma tarefa trivial, pois quanto melhor o valor da função objetivo, melhor é a solução. Na otimização multiobjetivo, a avaliação da fronteira de Pareto não é uma tarefa trivial. De acordo com Zitzler, Deb e Thiele (2000), a distância entre as fronteiras obtida e real deve ser minimizada, as soluções devem ter uma boa distribuição na fronteira e a fronteira deve ser extensa, dominando um amplo conjunto de soluções.

De forma similar, para Li e Yao (2019), convergência, dispersão, uniformidade e cardinalidade são aspectos de qualidade para avaliar a fronteira de Pareto. A convergência mede o quão próximas estão as fronteiras obtida e real, a dispersão mede a cobertura da fronteira, a uniformidade está relacionada à distribuição das soluções na fronteira e a cardinalidade ao número de soluções na fronteira.

Há diversos indicadores de qualidade que mapeiam a qualidade da fronteira de Pareto para um número real. Os indicadores de qualidade que incorporam os quatro aspectos listados acima são os mais comumente utilizados. Eles podem ser classificados quanto à metodologia empregada em (LI; YAO, 2019):

- ❑ **Indicadores baseados em distância:** requerem a fronteira de Pareto real, ou uma aproximação da fronteira real, para o cálculo de distância com a fronteira em avaliação. Fronteiras muito próximas terão boa avaliação nesta classe de indicadores. *Inverted Generational Distance* (IGD), *Dist1* e variantes de IGD são exemplos destes indicadores.

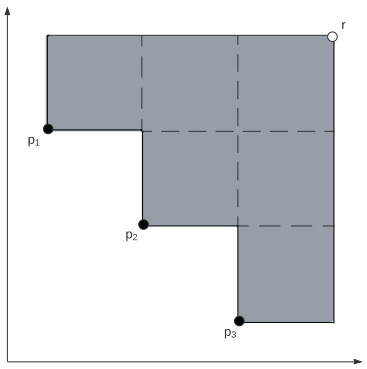
- ❑ **Indicadores baseados em volume:** calculam o volume entre a fronteira obtida e alguma especificação, como um ponto de referência por exemplo. Hipervolume, *R Family* e *Integrated Preference Functional* (IPF) são exemplos destes indicadores.

Segundo Zitzler e Thiele (1998), o hipervolume é o indicador de qualidade mais popular. Ele não requer conhecimento prévio da fronteira de Pareto real, podendo ser aplicado em muitos problemas de otimização multiobjetivo reais. Dados a fronteira de Pareto A e um ponto de referência r, o hipervolume de A é a região dominada por A e limitada por um ponto de referência r. A Equação 9 apresenta a fórmula do hipervolume, onde λ é a medida de Lebesgue (LI; YAO, 2019).

$$Hipervolume(A) = \lambda(\bigcup_{a \in A} \{x | a \prec x \prec r\}) \tag{9}$$

O hipervolume de A é entendido como o volume da união dos hipercubos determinados por cada ponto (solução) da fronteira de Pareto e o ponto de referência r. Na Figura 10 o cálculo do hipervolume é exemplificado para um problema de minimização com dois objetivos. Os hipercubos de cada ponto da fronteira ($\{p_1, p_2, p_3\}$) ao ponto r estão em destaque e o hipervolume é o volume da união dos hipercubos. Considerando duas fronteiras de Pareto A e B, diz-se que a fronteira A é melhor que a fronteira B, quando o hipervolume de A é maior que o hipervolume de B.

Figura 10 – Hipervolume para um problema de minimização com dois objetivos.



Fonte: Fonseca, Paquete e López-Ibánez (2006).

Para o cálculo do hipervolume, o ponto de referência deve ser definido. Não há um consenso sobre como definir o ponto de referência adequado para um problema, apesar

de o ponto *nadir* (ponto com os piores valores dos objetivos) ser bastante empregado em aplicações práticas (LI; YAO, 2019). Além disso, a complexidade computacional para o cálculo do hipervolume cresce exponencialmente com o número de objetivos, sendo, portanto, uma medida mais aplicada em problemas com dois ou três objetivos.

2.4 Considerações Finais

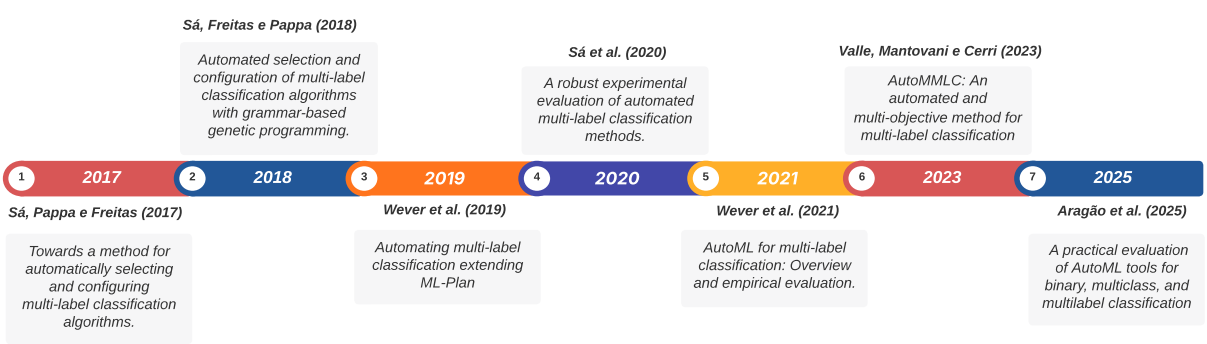
Este capítulo abordou os fundamentos teóricos essenciais para a pesquisa, contemplando os três temas principais: (1) os princípios da classificação multirrótulo, (2) os conceitos e técnicas do AutoML, e (3) os conceitos e algoritmos da otimização multiobjetivo. No próximo capítulo, será realizada uma revisão dos trabalhos relacionados na literatura.

Capítulo 3

Trabalhos Relacionados

Para o embasamento teórico sobre AutoML e problemas multi-alvo, conduzimos uma revisão sistemática da literatura com o objetivo de analisar e identificar abordagens e algoritmos comumente empregados, bem como conhecer as tendências nessa área de pesquisa (VALLE; MANTOVANI; CERRI, 2023b). No entanto, este capítulo não apresenta a revisão sistemática em sua totalidade, mas apenas os estudos que exploram o AutoML aplicado à classificação multirrótulo, especificamente quando abordados como um problema CASH. O capítulo também inclui dois novos trabalhos que não estavam na revisão sistemática. A Figura 11 ilustra os trabalhos relacionados em uma linha do tempo. No total, são sete trabalhos, publicados entre 2017 e 2025. Esses trabalhos são revisados nas Seções 3.1 a 3.7, seguidas por uma síntese das características mais relevantes de cada um (Seção 3.8).

Figura 11 – Linha do tempo dos trabalhos relacionados.



3.1 Seleção Automática de Algoritmos de Classificação Multirrótulo

Sá, Pappa e Freitas (2017) propuseram uma ferramenta AutoML denominada *GA-Auto-MLC*. *GA-Auto-MLC* seleciona automaticamente o melhor algoritmo de classificação multirrótulo (com uma configuração de hiperparâmetros) para um conjunto de dados de entrada. Os autores estruturaram o hiperespaço hierarquicamente, incluindo 31 algoritmos de classificação multirrótulo e nove de classificação monorrótulo.

Para garantir que os algoritmos de classificação multirrótulo tivessem configuração de hiperparâmetros corretas, Sá, Freitas e Pappa (2018) representaram o hiperespaço usando uma gramática livre de contexto. Uma gramática descreve as possíveis restrições e dependências sobre os algoritmos de AM e seus hiperparâmetros. Segundo os autores, uma gramática G é uma quádrupla $< N, T, P, S >$, onde N é o conjunto de não-terminais, T é o conjunto de terminais, P é o conjunto de regras de produção e S é o símbolo não-terminal inicial. A derivação da gramática, partindo do símbolo não-terminal inicial (S) e aplicando regras de produção (P), gera soluções candidatas válidas (algoritmo de classificação e configuração de hiperparâmetros) para um problema de AM.

Na ferramenta *GA-Auto-MLC*, Sá, Pappa e Freitas (2017) buscaram o melhor algoritmo de classificação multirrótulo com sua configuração de hiperparâmetros utilizando Algoritmos Evolutivos. Os autores representaram um indivíduo como um vetor de números reais de tamanho 15, com valores dentro do intervalo $[0, 1]$. Um indivíduo representa um algoritmo de classificação e sua respectiva configuração de hiperparâmetros. Um modelo de classificação é treinado para cada indivíduo gerado no processo evolutivo, e posteriormente avaliado pela função de *fitness*. É importante salientar que os melhores indivíduos têm maior probabilidade de serem selecionados para participar do *crossover* e da mutação, para gerar novas populações.

A medida de avaliação utilizada como *fitness* pelo algoritmo evolutivo é uma medida composta por quatro métricas de desempenho, conforme a Equação 10. Na equação, EM denota a medida de avaliação *Exact Match*, HL é *Hamming Loss*, FM a medida *F-score* Macro e RL a medida *Ranking Loss*.

$$fitness = \frac{EM + (1 - HL) + FM + (1 - RL)}{4} \tag{10}$$

Para fins de comparação, os autores utilizaram dois algoritmos de classificação multirrótulo como *baselines*: BR e CC. Ambos foram implementados tendo o algoritmo *Support Vector Machine* (SVM) como classificador base, cujos hiperparâmetros γ e C foram ajustados por meio de uma Busca em Grade. Tanto o *GA-Auto-MLC* como os *baselines* foram executados em três conjuntos de dados da biblioteca `MULAN`, e avaliados pelas quatro medidas de desempenho acima citadas. Em 24 casos de testes, *GA-Auto-MLC* superou

estatisticamente os *baselines* 10 vezes, foi significativamente superado pelos *baselines* cinco vezes e não apresentou diferença significativa em relação aos *baselines* nove vezes.

3.2 AutoML para Classificação Multirrótulo usando Programação Genética baseada em Gramática

Sá, Freitas e Pappa (2018) também exploraram gramáticas para representar o hiperespaço e trabalharam com duas versões de sistemas AutoML: *Auto-*$MEKA_{GGP(S)}$ e *Auto-*$MEKA_{GGP}$. O primeiro utiliza uma gramática simplificada baseada no hiperespaço do algoritmo *GA-Auto-MLC* (SÁ; PAPPA; FREITAS, 2017). Já o hiperespaço da versão *Auto-*$MEKA_{GGP}$ era composto por um total de 60 algoritmos – 30 algoritmos de classificação multirrótulo e 30 algoritmos de classificação monorrótulo. Também foram incluídos no hiperespaço os hiperparâmetros desses algoritmos.

No trabalho supracitado, os autores utilizaram Programação Genética para encontrar o melhor algoritmo de classificação multirrótulo e sua respectiva configuração de hiperparâmetros. Nos experimentos, os autores representaram o hiperespaço por meio de uma gramática, adotando no processo de otimização tanto a programação genética com gramática simplificada (*Auto-*$MEKA_{GGP(S)}$) quanto com gramática completa (*Auto-*$MEKA_{GGP}$). A função de *fitness* foi a mesma adotada por Sá, Pappa e Freitas (2017) (Equação 10). Cada indivíduo foi obtido da derivação das regras de produção da gramática, formando uma árvore. Os operadores de *crossover* e mutação de Whigham (MCKAY et al., 2010), que modificam os indivíduos, também foram guiados pela gramática.

Como *baselines* para comparações, os autores utilizaram o algoritmo genético proposto por Sá, Pappa e Freitas (2017) e os algoritmos de classificação multirrótulo BR e CC. Para esses dois últimos, os autores utilizaram Busca Local para definir o melhor algoritmo de classificação monorrótulo a ser adotado dentre os 11 algoritmos avaliados, todos com seus hiperparâmetros padrão (*default*). Todos os algoritmos foram testados em 10 conjuntos de dados do repositório *Knowledge and Discovery Systems* (KDIS) e comparados quanto à função *fitness*. Os resultados obtidos pelos autores demonstraram que *Auto-*$MEKA_{GGP}$ alcançou a maior precisão preditiva em seis conjuntos de dados, enquanto *Auto-*$MEKA_{GGP(S)}$ e *GA-Auto-MLC* tiveram resultados melhores em dois conjuntos de dados cada.

3.3 Extensão da Ferramenta ML-Plan para o Cenário Multirrótulo

Wever et al. (2019) empregaram Redes de Tarefas Hierárquicas (do inglês *Hierarchical Task Network* (HTN)) (GEORGIEVSKI; AIELLO, 2015) na representação da natureza hi-

erárquica dos hiperespaços. De acordo com os autores, essas redes permitem decompor tarefas complexas em outras mais simples ou igualmente complexas. A decomposição ocorre até que haja apenas tarefas simples. Wever et al. (2019) estenderam a ferramenta *ML-Plan* (MOHR; WEVER; HüLLERMEIER, 2018; WEVER; MOHR; HÜLLERMEIER, 2018), originalmente desenvolvida para modelar hiperespaços de classificação monorrótulo com Redes de Tarefas Hierárquicas. Os autores criaram uma nova ferramenta chamada ML^2-$PLAN$. O novo hiperespaço incluiu algoritmos para classificação monorrótulo e multirrótulo, e o espaço de hiperparâmetros desses algoritmos.

O hiperespaço foi representado por meio de um grafo, onde o algoritmo de busca *Best First* foi executado de forma a minimizar o erro da classificação. Inicialmente, o erro atribuído aos nós relacionados aos classificadores multirrótulo era zero. Esse valor foi atualizado durante a busca. O objetivo era que cada algoritmo multirrótulo fosse selecionado ao menos uma vez durante o processo de busca. A função empregada como objetivo da otimização foi a mesma adotada por Sá, Pappa e Freitas (2017) (Equação 10).

Nos experimentos realizados pelos autores, o ML^2-$Plan$ foi comparado com três *baselines* que foram ajustados os algoritmos de otimização: 1) Busca Aleatória; 2) Algoritmo Genético (SÁ; PAPPA; FREITAS, 2017); e 3) Programação Genética baseada em Gramática (SÁ; FREITAS; PAPPA, 2018). Os algoritmos foram executados em 14 conjuntos de dados da biblioteca `MULAN` e comparados quanto à função definida na Equação 10. Segundo Wever et al. (2019), o ML^2-$PLAN$ apresentou resultados superiores aos *baselines* em 10 dos 14 conjuntos de dados, demonstrando resultados sólidos e desempenho significativamente melhor que os *baselines*.

3.4 Avaliação Experimental de Métodos AutoML para Classificação Multirrótulo

Dando sequência a seus trabalhos anteriores (SÁ; PAPPA; FREITAS, 2017; SÁ; FREITAS; PAPPA, 2018), Sá et al. (2020) propuseram dois novos sistemas AutoML:

- *Auto-*$MEKA_{spGGP}$: uma extensão do *Auto-*$MEKA_{GGP}$ (SÁ; FREITAS; PAPPA, 2018), que especializa a Programação Genética baseada em Gramática para aumentar a diversidade de soluções; e
- *Auto-*$MEKA_{BO}$: um sistema AutoML baseado na Otimização Bayesiana.

Os dois sistemas AutoML propostos foram comparados com o *Auto-*$MEKA_{GGP}$ (SÁ; FREITAS; PAPPA, 2018) e com os *baselines* AutoML baseados na Busca Aleatória (*Auto-*$MEKA_{RS}$) e na Busca Gulosa (*Auto-*$MEKA_{GS}$).

Os sistemas AutoML do experimento foram comparados usando hiperespaços de diferentes tamanhos, considerando diferentes quantidades de algoritmos de classificação

monorrótulo e multirrótulo. Para isso, foram definidos três hiperespaços: pequeno (10 algoritmos), médio (30 algoritmos) e grande (54 algoritmos). O objetivo foi verificar a influência dos hiperespaços na acurácia dos classificadores multirrótulo, uma vez que quanto mais algoritmos, hiperparâmetros e rótulos, mais complexo torna-se o problema AutoML.

Quatorze conjuntos de dados do repositório KDIS foram utilizados nos experimentos. As predições resultantes das execuções dos sistemas AutoML foram avaliadas por meio das medidas *Hamming Loss*, *Ranking Loss* e da função de *fitness* definida na Equação 10. Segundo Sá et al. (2020), o *Auto-*$MEKA_{GGP}$ destacou-se como o melhor sistema AutoML dentre os avaliados. Quanto ao tamanho do hiperespaço, Sá et al. (2020) concluíram que hiperespaços menores (pequeno e médio) simplificam a busca e tornam os resultados dos algoritmos de otimização semelhantes. Enquanto hiperespaços maiores (grande) aumentam a exploração de possíveis configurações e, consequentemente, a complexidade da busca. Assim, algoritmos de otimização que lidam melhor com a exploração dos hiperespaços podem alcançar resultados melhores.

3.5 Visão Geral e Avaliação Empírica de AutoML para Classificação Multirrótulo

SegundoWever et al. (2021), o hiperespaço é um elemento essencial nas comparações entre trabalhos na área de AutoML. Embora seja comum comparar diferentes algoritmos de otimização, nem sempre os hiperespaços empregados são os mesmos. Essa divergência torna a interpretação dos resultados mais complexa. Os autores destacam ainda que o hardware empregado, o tempo limite para a execução dos classificadores e a função de avaliação das soluções candidatas também podem interferir nos resultados. Como solução, os autores propuseram um *framework* AutoML para classificação multirrótulo, dividido em duas partes:

1. **Especificações Técnicas**: engloba restrições gerais de execução (hardware disponível e tempo limite para execução, por exemplo), a descrição do hiperespaço e as medidas de avaliação para as soluções candidatas; e

2. **Otimizador**: é responsável por conectar um algoritmo de otimização ao restante do *framework*. O otimizador traduz a descrição do hiperespaço para um formato próprio, coleta a medida de desempenho por solução candidata e gera uma resposta do *framework*.

Na avaliação do *framework*, Wever et al. (2021) estabeleceram especificações técnicas comuns a todos os otimizadores, incluindo os algoritmos de otimização. O hiperespaço continha 31 algoritmos de classificação multirrótulo, 36 algoritmos de classificação

monorrótulo e uma única opção de *kernel* para o algoritmo *Sequential Minimal Optimization* (SMO) (versão `WEKA` do algoritmo SVM). Diferentes medidas de avaliação foram adotadas, incluindo F-score baseado em instâncias e rótulos, em suas variantes macro e micro. As execuções foram feitas no mesmo equipamento, considerando as mesmas restrições de tempo de execução e 24 conjuntos de dados distintos. Foram avaliados os seguintes algoritmos de otimização: Otimização Bayesiana, Hiperbanda, Otimização Bayesiana em conjunto com Hiperbanda, Programação Genética baseada em Gramática, Busca Gulosa em Rede de Tarefas Hierárquicas e Busca Aleatória. Os melhores resultados do experimento foram encontrados com o algoritmo Busca Gulosa em Rede de Tarefas Hierárquicas.

3.6 AutoMMLC

Valle, Mantovani e Cerri (2023a) propuseram uma estratégia AutoML multiobjetivo para classificação multirrótulo, denominada *Automated Multi-objective strategy for Multi-Label Classification* (AutoMMLC). Essa estratégia foi um trabalho preliminar, base para o desenvolvimento desta pesquisa. AutoMMLC busca algoritmos de classificação multirrótulo e configuração de hiperparâmetros que otimizam tanto a medida de desempenho F-score baseada em instâncias quanto o tempo de treinamento dos classificadores. Assim, o objetivo da estratégia foi encontrar automaticamente modelos que maximizam o F-score e que apresentam menor tempo de treinamento.

A estratégia AutoMMLC foi desenvolvida em duas versões, uma com o algoritmo MORS e outra com o NSGA-II. Ambas as versões trabalhavam com um hiperespaço composto por 11 algoritmos de classificação monorrótulo e 10 algoritmos de classificação multirrótulo, das bibliotecas `scikit-learn` e `scikit-multilearn`, respectivamente. Os indivíduos do NSGA-II foram representados por meio de vetores de números inteiros, cujo tamanho era maior que o número de hiperparâmetros do hiperespaço, uma vez que para cada hiperparâmetro do hiperespaço era necessário um gene do indivíduo.

Os experimentos foram realizados com dez conjuntos de dados multirrótulo. As fronteiras de Pareto resultantes de AutoMMLC$_{MORS}$ e AutoMMLC$_{NSGA-II}$ foram avaliadas por meio do hipervolume. O teste de Wilcoxon demonstrou que as versões do AutoMMLC apresentaram resultados estatisticamente semelhantes para essa métrica.

Os algoritmos *baselines* do experimento foram BR, *BR and KNN in version A* (BRkNNa), *BR and KNN in version B* (BRkNNb), CC, *Decision Table* (DT), MLkNN, *Multi-Label Support Vector Machine* (MLTSVM), *Multi-Label Adaptive Resonance Associative Map* (MLARAM), *Random Forest* (RF) e *Random K-Label Disjoint Pruned Sets* (RAkELd). Para comparar as versões de AutoMMLC com os algoritmos *baselines*, Valle, Mantovani e Cerri (2023a) selecionaram uma solução de cada fronteira de Pareto utilizando o conceito de *Frugality Score* (EVCHENKO, 2016). As comparações foram feitas em relação ao tempo de treinamento e ao F-score. O teste de Friedman

demonstrou que os algoritmos selecionados das fronteiras de AutoMMLC$_{MORS}$ e de AutoMMLC$_{NSGA-II}$ apresentaram desempenhos iguais tanto para o F-score quanto para o tempo de treinamento. Além disso, apresentaram resultados superiores aos dos algoritmos *baselines* para o F-score e melhores do que a maioria dos *baselines* para tempo de treinamento.

3.7 Avaliação Prática de Ferramentas AutoML

Aragão et al. (2025) realizaram uma avaliação abrangente de 16 ferramentas AutoML aplicadas a problemas de classificação binária, multiclasse e multirrótulo. Como nosso interesse é na classificação multirrótulo, revisamos o trabalho focando nesse tipo de problema. No contexto da classificação multirrótulo, os autores abordaram o problema de forma nativa, utilizando ferramentas que suportam naturalmente os múltiplos rótulos, e também por meio da técnica *Label Powerset*, que transforma o problema multirrótulo em multiclasse pela criação de novas classes – resultantes das combinações dos rótulos originais. As ferramentas foram avaliadas em sete conjuntos de dados multirrótulo, com orçamento de tempo de cinco minutos por execução e avaliadas pela medida de desempenho F-score baseada em instância.

Entre as ferramentas que suportam nativamente a classificação multirrótulo estão o `Auto-sklearn`, `AutoKeras`, `AutoGluon` e `FEDOT` (*Framework for Evolutionary Design of Optimal Trees*). O `Auto-sklearn` destacou-se com os melhores resultados em termos de F-score, enquanto `FEDOT` apresentou desempenho intermediário. `AutoGluon` e `AutoKeras` tiveram desempenhos inferiores, com maior variabilidade entre execuções. Quanto ao tempo de execução, `Auto-sklearn` e `FEDOT` consumiram praticamente todo o orçamento de tempo. Com a transformação *Label Powerset*, 15 das 16 ferramentas apresentaram resultados. No entanto, houve conjuntos de dados para os quais algumas ferramentas não encontraram soluções. Mais uma vez, o `Auto-sklearn` encontrou resultados superiores em relação às demais ferramentas. A transformação dos rótulos aumentou drasticamente o número de classes, o que tornou o problema mais difícil, especialmente quando os conjuntos de dados apresentavam maior dimensionalidade de rótulos.

Os resultados demonstraram que há uma limitação do suporte nativo à classificação multirrótulo nas ferramentas AutoML avaliadas. Embora a técnica de *Label Powerset* amplie o conjunto de ferramentas utilizáveis, ela acarreta maior complexidade computacional e perda de desempenho. Dentre as ferramentas nativas, aquelas com estratégias de busca mais robustas – como a otimização Bayesiana – mostraram-se mais eficazes para lidar com os múltiplos rótulos, embora a um custo computacional mais elevado.

3.8 Síntese dos Trabalhos Relacionados

A fim de sintetizar as características dos trabalhos relacionados, respondemos às seguintes questões.

3.8.1 Como o hiperespaço é definido?

O hiperespaço compreende os algoritmos do AM e seus respectivos espaços de hiperparâmetros. Aragão et al. (2025) não definem hiperespaços de busca em seu trabalho, os autores testam as ferramentas `Auto-sklearn`, `AutoKeras`, `AutoGluon` e `FEDOT`, que definem seus próprios hiperespaços. Os demais trabalhos revisados neste capítulo definem hiperespaços de busca, eles são constituídos por classificadores monorrótulo e multirrótulo.

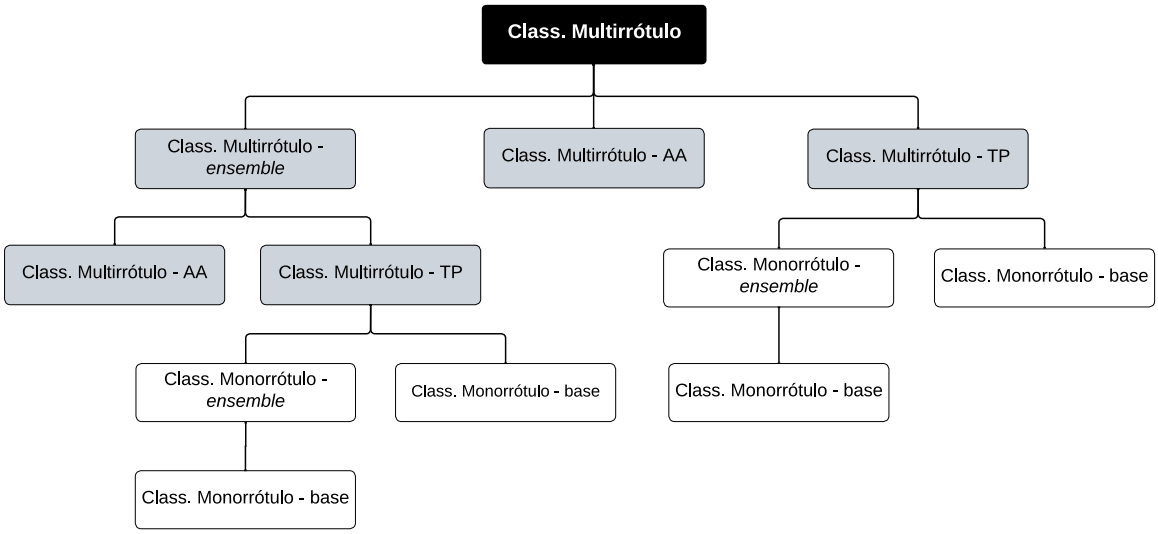


Figura 12 – Representação da hierarquia das categorias de algoritmos do hiperespaço de busca. Na Figura, AA e TP correspondem às abordagens adaptação de algoritmo e transformação de problema.

Os algoritmos monorrótulo tratam diretamente o problema monorrótulo (base) ou são *ensemble* de classificadores. Os algoritmos de classificação multirrótulo pertencem às abordagens adaptação de algoritmo, transformação de problema e algoritmos *ensemble*. O uso desses algoritmos estabelece uma relação hierárquica entre os algoritmos. A Figura 12 ilustra essa hierarquia de algoritmos. Para exemplificar, um algoritmo multirrótulo (Class. Multirrótulo) pode ser um *ensemble* de algoritmos (Class. Multirrótulo - *ensemble*) da abordagem transformação de problema (Class. Multirrótulo - TP), utilizando como classificador base um algoritmo de classificação monorrótulo (Class. Monorrótulo - base). Cada algoritmo dessa hierarquia possui seus respectivos hipereparâmetros.

Nos trabalhos relacionados, onde os algoritmos do hiperespaço são identificados, o hiperespaço é modelado de forma hierárquica. A Tabela 1 identifica as categorias de algoritmos de cada trabalho. O hiperespaço de Sá, Pappa e Freitas (2017), por exemplo,

é composto por algoritmos de classificação multirrótulo das abordagens adaptação de algoritmo, transformação de problema e *ensemble*, além de algoritmos de classificação monorótulo (base).

Na maioria dos trabalhos, as implementações disponíveis no hiperespaço são algoritmos das bibliotecas `WEKA` e `MEKA`. Apenas Valle, Mantovani e Cerri (2023a) utilizaram algoritmos das bibliotecas `scikit-learn` e `scikit-multilearn`. Quanto aos hiperparâmetros, a análise dos hiperespaços demonstrou que Sá, Pappa e Freitas (2017), Sá, Freitas e Pappa (2018), Sá et al. (2020) e Wever et al. (2021) representaram os hiperparâmetros de um algoritmo praticamente na sua totalidade. Já Valle, Mantovani e Cerri (2023a) representaram apenas os hiperparâmetros mais relevantes de um algoritmo.

Tabela 1 – Composição dos hiperespaços dos trabalhos relacionados. Na tabela, AA e TP correspondem às abordagens adaptação de algoritmo e transformação de problema.

	Referência	**Biblioteca**	**Class. Monorrótulo**		**Class. Multirrótulo**		
			Base	**Ensemble**	**AA**	**TP**	**Ensemble**
1	Sá, Pappa e Freitas (2017)	`WEKA`, `MEKA`	✓	-	✓	✓	✓
2	Sá, Freitas e Pappa (2018)	`WEKA`, `MEKA`	✓	-	✓	✓	✓
3	Wever et al. (2019)*	`WEKA`, `MEKA`	-	-	-	-	-
4	Sá et al. (2020)	`WEKA`, `MEKA`	✓	-	✓	✓	-
5	Wever et al. (2021)	`WEKA`, `MEKA`	✓	✓	✓	✓	✓
6	Valle, Mantovani e Cerri (2023a)	`scikit-learn`, `scikit-multilearn`	✓	✓	✓	✓	✓

* Este trabalho não menciona os algoritmos utilizados.

3.8.2 Quais algoritmos de otimização são empregados?

Sá, Pappa e Freitas (2017), Sá, Freitas e Pappa (2018), Wever et al. (2019), Sá et al. (2020), Wever et al. (2021) empregam algoritmos de otimização mono-objetivo em suas abordagens AutoML, como Algoritmos Genéticos, Hiperbanda e Otimização Bayesiana. Além disso, na maioria desses trabalhos, os algoritmos *baselines* foram estratégias AutoML que também empregam algoritmos de otimização. Valle, Mantovani e Cerri (2023a) utilizaram o algoritmo multiobjetivo NSGA-II. Aragão et al. (2025) não propuseram abordagens AutoML novas, os autores avaliaram ferramentas prontas, que adotam diferentes algoritmos de otimização mono-objetivo. A Tabela 2 sumariza os algoritmos de otimização utilizados nos trabalhos relacionados.

Tabela 2 – Resumo dos algoritmos de otimização utilizados nos trabalhos relacionados. Esses algoritmos referem-se à proposta principal dos trabalhos e também aos algoritmos de otimização empregados nos *baselines* de comparação.

	Refências	**Algoritmo de Otimização no AutoML**	**Algoritmo de Otimização dos *Baselines***
1	Sá, Pappa e Freitas (2017)	Algoritmo Genético	-
2	Sá, Freitas e Pappa (2018)	Programação Genética baseada em Gramática	Algoritmo Genético
3	Wever et al. (2019)	Busca Gulosa em Rede de Tarefas Hierárquicas	Busca Aleatória, Algoritmo Genético e Programação Genética baseada em Gramática
4	Sá et al. (2020)	Programação Genética baseada em Gramática, Otimização Bayesiana	Programação Genética baseada em Gramática, Busca Aleatória e Busca Gulosa
5	Wever et al. (2021)	Hiperbanda, Otimização Bayesiana, Otimização Bayesiana em conjunto com Hiperbanda	Programação Genética baseada em Gramática, Busca Gulosa em Rede de Tarefas Hierárquicas e Busca Aleatória
6	Valle, Mantovani e Cerri (2023a)	NSGA-II	-
7	Aragão et al. (2025)	Não há propostas de novas abordagens AutoML	`Auto-sklearn` (Otimização Bayesiana), `AutoKeras` (Otimização Bayesiana e NAS), `AutoGluon` (Busca Aleatória e Hiperbanda) e `FEDOT` (Algoritmos Evolucionários)

3.8.3 Como as soluções candidatas são avaliadas durante a otimização?

Nos estudos de Sá, Pappa e Freitas (2017), Sá, Freitas e Pappa (2018), Wever et al. (2019) e Sá et al. (2020), a avaliação das soluções candidatas foi realizada pela função da Equação 10. Essa equação determina uma medida composta pela média de quatro métricas de desempenho. Por sua vez, Wever et al. (2021) avaliaram seu *framework* com métricas como F-score por instâncias e por rótulos. Valle, Mantovani e Cerri (2023a) utilizaram o tempo de treinamento em conjunto com a medida F-score baseada em instâncias. Já Aragão et al. (2025) executaram ferramentas AutoML prontas, com a medida F-score baseada em instâncias. Dessa forma, não há um consenso sobre qual métrica ou conjunto de métricas deve ser utilizado para avaliar as soluções candidatas durante a otimização.

3.8.4 Há alguma técnica adicional que contribui com o AutoML?

Técnicas avançadas como o meta-aprendizado e a transferência de aprendizado (Seção 2.2.2) são potencialmente úteis para o AutoML. No entanto, nos trabalhos revisados em que novos sistemas AutoML são propostos, essas técnicas não foram empregadas. Aragão et al. (2025) também não as utilizaram diretamente. Ainda assim, as ferramentas

AutoML avaliadas por esses autores podem fazer uso dessas técnicas. O `auto-sklearn`, por exemplo, emprega o meta-aprendizado para sugerir configurações iniciais promissoras (FEURER; SPRINGENBERG; HUTTER, 2015).

3.8.5 Quais as limitações e propostas de trabalhos futuros?

Todos os trabalhos revisados neste capítulo, que propuseram estratégias AutoML para classificação multirrótulo, automatizaram apenas a tarefa de indução do modelo preditivo, e negligenciaram as demais tarefas do *pipeline*. Sá, Pappa e Freitas (2017) e Sá, Freitas e Pappa (2018) sugeriram como direções futuras a automação de tarefas relacionadas ao pré-processamento e ao pós-processamento de dados, embora não tenham desenvolvido nada nesse sentido.

Para a tarefa de indução dos modelos preditivos, as limitações e propostas futuras concentram-se nas medidas de avaliação e na definição do hiperespaço. De acordo com Sá et al. (2020), é preciso entender quais medidas de avaliação são mais adequadas ao AutoML, já que algumas são neutras durante o processo de otimização, produzindo resultados estáveis. Wever et al. (2019) destacam que a otimização multiobjetivo poderia ser aplicada com diferentes medidas de desempenho.

As diferentes combinações de algoritmos e hiperparâmetros geram um hiperespaço grande e difícil de ser explorado, considerando a complexidade de tempo e de espaço. Para Sá, Pappa e Freitas (2017) e Wever et al. (2021), é necessário melhorar a definição dos hiperespaços, pois os algoritmos de otimização encontram dificuldades para lidar com a sua alta dimensão. Sá et al. (2020) argumentam que é necessário entender quais algoritmos devem compor o hiperespaço, pois as buscas seriam feitas em um espaço menor, com os algoritmos de classificação multirrótulo mais promissores. Wever et al. (2019) sugerem melhorar a escalabilidade do hiperespaço por meio da adoção de arquiteturas orientadas a serviços ou de técnicas de meta-aprendizagem, embora não tenham aplicado o meta-aprendizado em seus experimentos. Similarmente, Wever et al. (2021) citam abordagens de meta-aprendizagem para iniciar a busca com soluções candidatas mais promissoras, tornando a busca mais eficiente.

3.9 Considerações Finais

Neste capítulo, revisamos os principais trabalhos relacionados ao tema desta tese, com foco em AutoML para classificação multirrótulo (abordagem CASH). A revisão contextualiza o estado da arte, identificando potenciais lacunas e oportunidades de pesquisa que motivaram a proposta deste trabalho. Os trabalhos analisados serviram como base teórica e prática para o desenvolvimento desta pesquisa.

Capítulo 4

Metodologia Experimental

Este capítulo descreve as principais escolhas experimentais e metodológicas adotadas ao longo da tese. Trata-se de uma metodologia geral, descrita de forma a evitar a repetição dos mesmos assuntos nos capítulos dedicados a cada experimento. As próximas seções deste capítulo estão organizadas da seguinte forma: as Seções 4.1 a 4.4 apresentam os conjuntos de dados, os conjuntos de dados pré-processados, as medidas de avaliação e o *setup* dos algoritmos de otimização; a Seção 4.5 lista os *baselines* adotados para comparação, bem como seus hiperparâmetros; a Seção 4.6 descreve os testes estatísticos adotados para validação dos resultados experimentais; a Seção 4.7 lista os repositórios com os códigos dos experimentos; e por fim, a Seção 4.8 apresenta as considerações finais do capítulo.

4.1 Conjuntos de Dados

Os experimentos reportados nesta tese foram executados em um total de 14 conjuntos de dados para classificação multirrótulo. A Tabela 3 lista estes conjuntos e também apresenta características gerais destes problemas, como: seus domínios, número de atributos, número de rótulos, número de instâncias, cardinalidade (média do número de rótulos de uma instância) e densidade (média de rótulos relevantes por instância em relação ao total de rótulos). Todos os conjuntos de dados foram obtidos do repositório `MULAN`[1]. Entretanto, é importante mencionar algumas alterações em relação às versões originais: i) o primeiro atributo do conjunto de dados *genbase* foi removido, pois ele é uma identificação das instâncias; ii) o conjunto de dados *tmc2007-500* é uma versão reduzida do conjunto de dados *tmc2007*, obtida após a seleção de atributos.

[1] Disponível em: <https://mulan.sourceforge.net/datasets-mlc.html>

Tabela 3 – Conjuntos de dados multirrótulo.

#	Conj. de Dados	Domínio	Atributos	Rótulos	Instâncias	Cardinalidade	Densidade
1	bibtex	texto	1836	159	7395	2.402	0.015
2	birds	áudio	260	19	645	1.014	0.053
3	cal500	música	68	174	502	26.044	0.150
4	corel5k	imagens	499	374	5000	3.522	0.009
5	delicious	texto	500	983	16105	19.020	0.019
6	emotions	música	72	6	593	1.869	0.311
7	enron	texto	1001	53	1702	3.378	0.064
8	flags	imagens	19	7	194	3.392	0.485
9	genbase	biologia	1185	27	662	1.252	0.046
10	mediamil	vídeo	130	101	43907	4.376	0.043
11	medical	texto	1449	45	978	1.074	0.028
12	scene	imagens	294	6	2407	1.074	0.179
13	tmc-2007-500	texto	500	22	28596	2.220	0.101
14	yeast	biologia	103	14	2417	4.237	0.303

Durante a realização dos experimentos e validação experimental dividiu-se os conjuntos de dados em 3 partições, por meio de validação cruzada estratificada. Além disso, as partições destinadas ao treinamento foram subdivididas utilizando o método *holdout*, reservando 70% dos dados para o treinamento e 30% para a validação. Adotamos $k = 3$ para evitar o aumento da complexidade das estratégias AutoML. A estratificação iterativa proposta por Sechidis, Tsoumakas e Vlahavas (2011) foi utilizada para assegurar a mesma proporção de instâncias de cada classe nas partições de dados. A estratificação foi codificada utilizando a biblioteca `scikit-multilearn` (SZYMANSKI; KAJDANOWICZ, 2019) e os conjuntos de dados resultantes foram salvos em arquivos com extensão *Attribute-Relation File Format* (ARFF). Para conjuntos de dados densos, as partições também foram salvas com dados normalizados.

4.2 Conjuntos de Dados Pré-processados

Alguns experimentos utilizaram versões destes conjuntos após pré-processamento. Dos 14 conjuntos de dados listados na Seção 4.1, 11 foram pré-processados de forma manual. Inicialmente, eliminamos rótulos associados a poucas instâncias (rótulos com dez ou menos instâncias) e atributos constantes (com os mesmos valores para todas as instâncias). Em seguida, cada conjunto foi subdividido usando validação cruzada estratificada com *3-folds* (SECHIDIS; TSOUMAKAS; VLAHAVAS, 2011). Por fim, para cada *fold*, eliminamos instâncias não rotuladas e duplicadas. As tarefas de pré-processamento foram codificadas utilizando as bibliotecas `pandas` (MCKINNEY, 2010) e `scikit-multilearn` (SZYMANSKI; KAJDANOWICZ, 2019). Os conjuntos de dados pré-processados também foram salvos em arquivos no formato ARFF e, caso gerassem arquivos densos, foram salvos em duas versões: com e sem normalização.

A Tabela 4 lista estatísticas gerais comparando os conjuntos de dados antes e após

Tabela 4 – Conjuntos de dados multirrótulo pré-processados.

		Antes do Pré-processamento		**Após o Pré-processamento**	
#	**Conjunto de Dados**	**Atributos**	**Rótulos**	**Atributos**	**Rótulos**
1	bibtex	1836	159	1836	159
2	birds	260	19	260	15
3	corel5k	499	374	499	218
4	cal500	68	174	68	141
5	emotions	72	6	72	6
6	enron	1001	53	1001	42
7	flags	19	7	19	7
8	genbase	11858	27	112	16
9	medical	1449	45	1449	20
10	scene	294	6	294	6
11	yeast	103	14	103	14

a etapa de pré-processamento. No geral não ocorreram muitas mudanças significativas. O número de atributos no conjunto de dados *genbase* diminuiu de 1185 para 112. Essa redução ocorreu devido ao elevado número de atributos com valores constantes. Os demais conjuntos de dados não tiveram alterações no número de atributos. Os conjuntos de dados *birds*, *corel5k*, *enron*, *genbase* e *medical* tiveram uma redução no número de rótulos, pois esses conjuntos possuíam rótulos com dez ou menos instâncias.

4.3 Medidas de Avaliação

Duas medidas de desempenho foram utilizadas para avaliar os algoritmos de classificação multirrótulo nos experimentos: Macro F-score, e o tamanho do modelo induzido. A métrica Macro F-score foi descrita na Seção 2.1.1, e o tamanho do modelo é o próprio tamanho do modelo induzido e serializado em bytes. Essas medidas foram utilizadas pois permitem avaliar o desempenho preditivo médio e a complexidade dos modelos induzidos.

No experimento sobre uso de AutoML para seleção de atributos, utilizando otimização mono-objetivo, a medida empregada foi Macro F-score (Capítulo 7). Ela foi utilizada como valor de aptidão *fitness*) durante a busca, bem como para avaliar a solução final retornada pela otimização. Os demais experimentos abordaram AutoML com otimização multiobjetivo, onde ambas as medidas foram usadas para calcular a aptidão durante a busca e também para avaliar os modelos selecionados da fronteira de Pareto (Capítulos 5, 6, 8 e 9). A medida utilizada para avaliar os modelos substitutos (*surrogate*) foi o coeficiente de determinação (R^2), essa medida fornece um indicativo da qualidade do ajuste do regressor (MILES, 2005; DANGETI, 2017) (Apêndice C).

4.4 *Setup* dos Algoritmos de Otimização

Os algoritmos de otimização utilizados na tese foram a Otimização Bayesiana (OB) e o NSGA-II (Seção 2.3). A OB é um método de otimização eficiente, uma vez que a função de aquisição orienta a busca para regiões do espaço de soluções com maior probabilidade de apresentar configurações de alto desempenho (BROCHU; CORA; FREITAS, 2010). Já o NSGA-II é um algoritmo amplamente utilizado para resolver problemas complexos do AM, sendo um padrão de referência entre os algoritmos evolutivos multiobjetivo empregados (DEB et al., 2002).

Nesta seção, descrevemos os hiperparâmetros apenas do NSGA-II, pois a OB foi utilizada apenas no experimento descrito no Capítulo 7, onde seus hiperparâmetros são definidos. Para os experimentos com NSGA-II, utilizamos a implementação disponível no biblioteca `Pymoo`, um *framework* de otimização em `Python` (BLANK; DEB, 2020). O operador de *crossover* empregado foi o *crossover* uniforme, cujos indivíduos descendentes têm 50% dos genes de cada um dos progenitores. A mutação utilizada foi a pontual, alterando apenas um dos genes do indivíduo com probabilidade de 5%.

O tamanho da população de indivíduos e o número máximo de gerações foram definidos como 100. Essa configuração foi adotada, pois fornece um total de 10.000 avaliações em uma mesma execução, valor mais que suficiente para que o algoritmo possa convergir. Vale ressaltar que com essa configuração o NSGA-II converge em todos os conjuntos de dados antes que esses limites pré-definidos sejam alcançados. Mesmo que haja um limite máximo de gerações (100), pode ocorrer a parada antecipada caso não haja alterações nos valores objetivos por 10 gerações consecutivas.

4.5 *Baselines* para Comparação dos Resultados

Os *baselines* adotados para comparação nos experimentos foram algoritmos e estratégias AutoML para classificação multirrótulo. As comparações foram feitas em relação aos objetivos da otimização (Macro F-score e/ou tamanho do modelo). Cada experimento adotou um subconjunto dos algoritmos listados nesta seção. Incluímos também os algoritmos de classificação multirrótulo, pois de acordo com García-Pedrajas et al. (2024) e Bogatinovski et al. (2022), eles estão entre os algoritmos de classificação multirrótulo com os melhores desempenhos em diferentes medidas de avaliação.

- ❏ ***Binary Relevance* (BR) com AdaBoost**: García-Pedrajas et al. (2024) incluem "Real AdaBoost.MH" como um dos algoritmos de classificação multirrótulo de melhor desempenho. Segundo os autores, ele é equivalente ao algoritmo BR com AdaBoost como classificador base. O BR transforma o conjunto de dados multirrótulo em q conjuntos de dados binários e utiliza um classificador monorrótulo (TSOUMAKAS; KATAKIS; VLAHAVAS, 2009), neste caso, o AdaBoost.

- ***Conditional Dependency Networks* (CDN)**: o algoritmo constrói um gráfico totalmente conectado, onde cada nó representa um rótulo y_i (GUO; GU, 2011). Um classificador binário é então treinado para cada nó i para prever y_i, tendo como atributos de entrada $\mathbf{X}$ e os rótulos dos nós adjacentes ao nó i.

- ***Instance-Based Logistic Regression for Multi-Label Classification* (IBLR+)**: o algoritmo combina o aprendizado baseado em instâncias com a regressão logística (CHENG; HÜLLERMEIER, 2009). IBLR+ induz modelos de regressão logística utilizando um conjunto de dados aumentado, onde os dados de entrada são os atributos das instância concatenados com os rótulos das instâncias vizinhas.

- ***Random k-labelsets Overlapping* (RAkELo)**: o algoritmo divide o espaço de rótulos em m subconjuntos de tamanho k e treina um classificador LP para cada subconjunto, construindo um comitê de classificadores (TSOUMAKAS; KATAKIS; VLAHAVAS, 2011). Um rótulo é atribuído a uma nova instância se mais da metade dos classificadores do comitê atribuíram esse rótulo à instância. RAkELo permite que um mesmo rótulo esteja em diferentes subconjuntos.

- ***Classifier Chains* (CC)**: este algoritmo constrói uma cadeia de classificadores binários, onde à medida que as classificações acontecem, os rótulos previstos são adicionados como novos atributos para as próximas classificações (READ et al., 2009). CC não está entre os algoritmos citados por García-Pedrajas et al. (2024) e Bogatinovski et al. (2022), mas o incluímos como *baseline* juntamente com sua versão ensemble.

- ***Ensembles of Classifier Chains* (ECC)**: este algoritmo é um ensemble de CC (READ et al., 2009).

A Tabela 5 lista os algoritmos *baselines* e suas correspondentes configurações de hiperparâmetros. Nas implementações desses algoritmos utilizamos as bibliotecas: `scikit-multilearn` (SZYMANSKI; KAJDANOWICZ, 2019), `MEKA` (READ et al., 2016) e `Mulan` (TSOUMAKAS et al., 2011). Os valores de hiperparâmetros foram definidos com base em García-Pedrajas et al. (2024). Os *baselines* foram executados em cada um dos 3 *folds* resultantes da validação cruzada estratificada iterativa. Tomou-se como resultado da execução o valor médio da medida de interesse (Macro F-score e/ou tamanho do modelo). `Auto-sklearn` e `Auto-MEKA` foram as estratégias AutoML utilizadas como *baselines*, e são descritas nas Seções 4.5.1 e 4.5.2.

Tabela 5 – Hiperparâmetros dos algoritmos de classificação multirrótulo adotados como *baselines*.

Algoritmo	Biblioteca	Hiperparâmetros
BR+AdaBoost	`scikit-multilearn`	Classificador base: AdaBoost (tamanho do ensemble 50)
CC	`MEKA`	Método para obter as cadeias: randômico
CDN	`MEKA`	Número de iterações: 1000 Número de iterações de coleção: 100 Classificador base: Random Forest (100 árvores)
ECC	`MEKA`	Tamanho do ensemble: 10
IBLR+	`MULAN`	Número de vizinhos: 10
RAkELo	`scikit-multilearn`	Número de classificadores: $2q$, onde q é o número de rótulos Número de rótulos para cada classificador: 3

4.5.1 Auto-sklearn

O `Auto-sklearn` é uma ferramenta AutoML baseada na biblioteca `scikit-learn` (FEURER et al., 2015). A ferramenta automatiza o pré-processamento de dados, o pré-processamento de atributos e a busca pelo modelo de classificação. Logo, a ferramenta busca a melhor configuração de algoritmos de pré-processamento de dados, de atributos e de classificação, utilizando a otimização Bayesiana mono-objetivo. No entanto, ela é flexível para habilitar ou desabilitar a automatização de tarefas relacionadas ao pré-processamento de dados e de atributos.

O `Auto-sklearn` não foi projetado especificamente para o cenário multirrótulo, mas tem suporte para este tipo de tarefa e por isso foi utilizado como um *baseline* neste trabalho. O hiperespaço de pré-processamento de atributos, mais especificamente a seleção de atributos, contém apenas o algoritmo SelectFromModel com ExtraTrees. Os demais algoritmos de seleção de atributos do hiperespaço são destinados para a classificação monorrótulo. Os algoritmos *Random Forest*, *Decision Trees* e uma estratégia equivalente ao algoritmo multirrótulo BR que treina um classificador por rótulo, compõem o hiperespaço de classificadores do `auto-sklearn`. Todos os algoritmos contidos no hiperespaço são algoritmos da biblioteca `scikit-learn`. A Tabela 6 apresenta os hiperparâmetros fixos utilizados para este *baseline*. Adotamos o Macro F-score como objetivo da otimização. Outros hiperparâmetros, como as tarefas a serem automatizadas, o tempo limite para a indução de um algoritmo de classificação multirrótulo, e o tempo limite de execução do `auto-sklearn`, foram configurados a depender do experimento.

Executamos o `auto-sklearn` nas partições de treino dos *3-folds* resultantes da validação cruzada estratificada iterativa e obtivemos um modelo de classificação multirrótulo para cada *fold*. O Macro F-score foi calculado utilizando as partições de teste. Para os experimentos em que o tamanho do modelo era importante, calculamos essa medida

de forma independente do AutoML, serializando o *pipeline* gerado pelo `auto-sklearn` e obtendo o tamanho em *bytes* do objeto serializado. No entanto, só podemos utilizar esse valor quando `auto-sklearn` automatiza apenas a busca pelo modelo de classificação. Para situações em que `auto-sklearn` automatiza o pré-processamento de atributos e a busca pelo modelo de classificação, o tamanho calculado não reflete apenas o tamanho do modelo de classificação.

Tabela 6 – Hiperparâmetros dos *baselines* AutoML.

Algoritmo	**Hiperparâmetros**
`Auto-MEKA`	Tamanho da população (`-N`): 100 Número de gerações (`-G`): 100 Reamostragem dos conjuntos de treinamento e validação após R gerações (`-R`): 5 Elitismo (`-E`): 5 Probabilidade de mutação (`-M`): 0.2 Probabilidade de *crossover* (`-X`): 0.8 Modo do hiperespaço (`-O`): 1 (tamanho médio)
`auto-sklearn`	Métrica (`metric`): f1_macro Número de configurações iniciais via meta-aprendizado (`initial_configurations_via_metalearning`): 0 Estratégia de reamostragem (`resampling_strategy`): holdout Argumentos da estratégia de reamostragem (`resampling_strategy_arguments`): tamanho do conjunto de treino 0.67 Configuração do ensemble (`ensemble_kwargs`): tamanho do ensemble 1 Número máximo de modelos salvos em disco: (`max_models_on_disc`): 1

4.5.2 Auto-MEKA

Sá, Pappa e Freitas (2017), Sá, Freitas e Pappa (2018) e Sá et al. (2020) estudaram AutoML para automatizar a busca pelo melhor algoritmo de classificação multirrótulo. Os autores avaliaram diferentes algoritmos de otimização mono-objetivo e o efeito do tamanho do hiperespaço no AutoML. Segundo Sá et al. (2020), dentre as estratégias AutoML avaliadas, a que usa programação genética baseada em gramática como método de busca foi a que obteve os melhores resultados. Essa estratégia foi denominada pelos autores como $\texttt{Auto-MEKA}_{\text{GGP}}$, e a adotamos como um *baseline* em nossos experimentos. No entanto, vamos nos referir a ele apenas como `Auto-MEKA`. Para mais informações sobre os trabalhos dos autores, consulte o Capítulo 3.

A Tabela 6 apresenta os hiperparâmetros fixos utilizados para `Auto-MEKA`, definidos com base em Sá, Freitas e Pappa (2018) e Sá et al. (2020). Outros hiperparâmetros, como o tempo para execução do algoritmo de classificação multirrótulo; e o tempo de execução de `Auto-MEKA`, foram definidos por experimento. O `Auto-MEKA` foi desenvolvido na linguagem de programação `Java`. Para sua execução, utilizamos o arquivo com extensão `jar` disponibilizado pelos autores. Executamos o `Auto-MEKA` nos *3-folds* e armazenamos os resultados das execuções. Conhecendo os melhores algoritmos de cada partição, repetimos treino e teste para o cálculo de Macro F-score e/ou do tamanho do modelo.

4.6 Testes Estatísticos

Os resultados experimentais também foram analisados por meio de testes estatísticos não-paramétricos. Nas comparações de algoritmos dois a dois usamos o teste estatístico de Wilcoxon (DEMŠAR, 2006; SANTAFE; INZA; LOZANO, 2015), com nível de significância de 95% ($\alpha = 0.05$). A hipótese nula definia que os dois algoritmos analisados apresentavam desempenhos equivalentes em uma determinada medida de desempenho.

Para a comparação de diferentes algoritmos em diferentes conjuntos de dados usamos o teste de Friedman (DEMŠAR, 2006; SANTAFE; INZA; LOZANO, 2015), com nível de significância de 95% ($\alpha = 0.05$). Nesse caso, a hipótese nula assegurava que os algoritmos analisados apresentavam desempenhos equivalentes. Caso a hipótese nula fosse rejeitada, aplicamos o teste *post-hoc* de Nemenyi para identificar quais algoritmos apresentavam diferenças estatisticamente significativas tendo como referência um valor de diferença crítica (CD).

Para as estratégias AutoML multiobjetivo, por exemplo, selecionamos algoritmos representativos da fronteira de Pareto para realizar o teste de Friedman. A hipótese nula assegurava que todos os algoritmos de classificação multirrótulo, encontrados por estratégias AutoML ou não, possuíam Macro F-score semelhantes. Caso a hipótese fosse rejeitada, o teste *post-hoc* de Nemenyi seria aplicado. Sob as mesmas premissas do teste de Friedman para Macro F-score, conduzimos também o teste estatístico para avaliar o tamanho dos modelos.

4.7 Repositórios

Os códigos desenvolvidos nos experimentos desta tese foram hospedados no GitHub. Todos os repositórios estão listados na Tabela 7.

Tabela 7 – Repositórios com as implementações desenvolvidas pelos autores.

#	Estratégia	Descrição	Repositório
1	EMANUEL	AutoML multiobjetivo	Link
2	EMANUEL$_{SM}$	AutoML multiobjetivo com modelos substitutos	Link
3	AutoMLFS	AutoML mono-objetivo incluindo seleção de atributos	Link
4	EMANUEL$_{FS}$	AutoML multiobjetivo incluindo seleção de atributos	Link
5	EMANUEL$_{FS+}$	AutoML multiobjetivo incluindo seleção de atributos e modelos substitutos	Link

4.8 Considerações Finais

Este capítulo descreve a metodologia experimental geral adotada na tese. Os conceitos descritos neste capítulo são aplicados nos experimentos descritos nos próximos capítulos.

Além disso, cada capítulo descreve questões metodológicas específicas de cada experimento, o que torna os experimentos completos e reprodutíveis.

Capítulo 5

EMANUEL

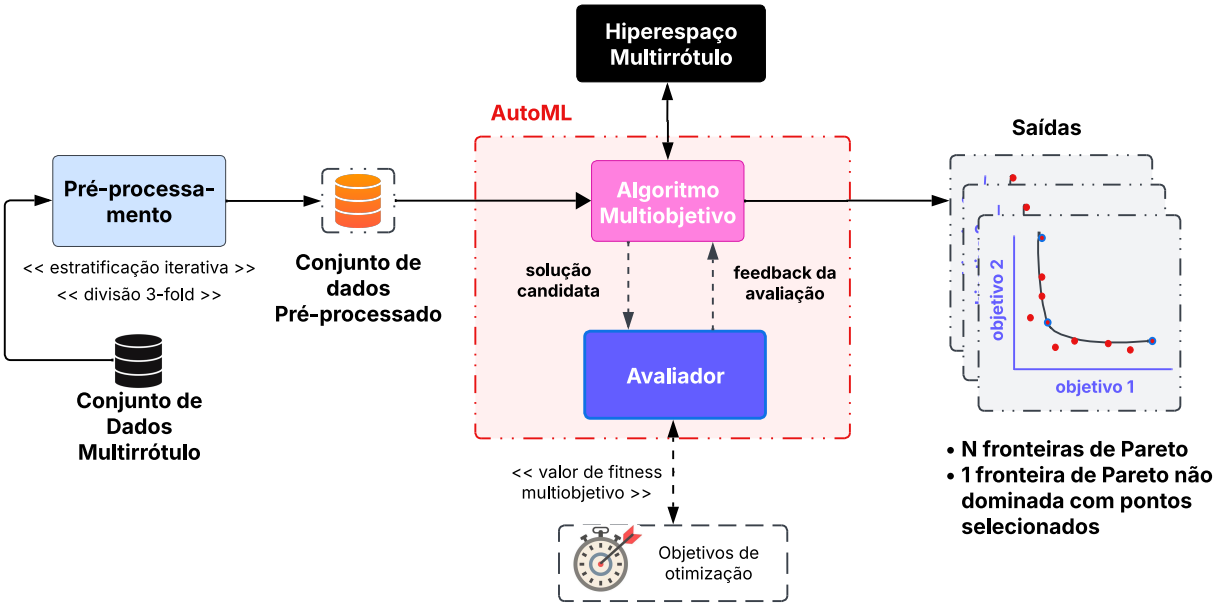
Diferentes algoritmos podem ser aplicados para resolver problemas de classificação multirrótulo, cada um com um desempenho preditivo distinto. Além disso, o desempenho de um algoritmo pode ser sensível aos hiperparâmetros configurados. Encontrar o algoritmo de classificação multirrótulo e a configuração de hiperparâmetros que otimiza uma ou mais medidas de desempenho é uma tarefa difícil. A busca por algoritmos de classificação multirrótulo ótimos, ou subótimos, pode ser automatizada por meio do AutoML.

No entanto, o AutoML ainda foi pouco explorado para problemas de classificação multirrótulo (SÁ et al., 2020; WEVER et al., 2021), especialmente quando a otimização envolve múltiplos objetivos. Assim, este capítulo propõe uma estratégia AutoML multiobjetivo para classificação multirrótulo, denominada *gEnetic Multi-objective strategy for the Automatic selectioN of mUlti-labEl cLassifiers* (EMANUEL), e investiga a sua aplicação. Nossa hipótese de pesquisa pressupõe que a nova estratégia pode encontrar um conjunto de soluções com desempenho competitivo em comparação com outros algoritmos existentes. As próximas seções deste capítulo estão organizadas da seguinte forma: a Seção 5.1 apresenta a estratégia proposta; a Seção 5.2 descreve os detalhes de implementação e de execução; a Seção 5.3 compara EMANUEL com os *baselines*; e a Seção 5.4 apresenta as considerações finais do capítulo.

5.1 Como EMANUEL Trabalha?

O funcionamento de EMANUEL é ilustrado na Figura 13. Inicialmente, um conjunto de dados multirrótulo passa pela etapa de pré-processamento. Nessa etapa, os dados são subdivididos utilizando validação cruzada estratificada *3-fold*. O conjunto de dados resultante do pré-processamento é entrada para o AutoML.

Figura 13 – Fluxograma do funcionamento da estratégia EMANUEL.



EMANUEL utiliza um algoritmo de otimização multiobjetivo. Esse algoritmo obtém do hiperespaço um algoritmo de classificação multirrótulo – uma solução candidata. O hiperespaço delimita quais são os algoritmos de classificação multirrótulo (incluindo hiperparâmetros) que a solução candidata pode assumir. A solução candidata é enviada ao avaliador, que treina e avalia o modelo induzido utilizando medidas referentes aos objetivos da otimização. O avaliador retorna um *feedback* da avaliação ao algoritmo de otimização. O processo de otimização ocorre de forma iterativa, até que uma condição de parada seja satisfeita.

O resultado de EMANUEL é uma fronteira de Pareto. Caso a estratégia seja executada N vezes, haverá N fronteiras de Pareto. Para encontrar uma fronteira resultante das N fronteiras originais, utilizamos o conceito de não dominância apresentado por Deb et al. (2002) e encontramos a primeira fronteira de Pareto não dominada. Essa fronteira contém vários algoritmos de classificação multirrótulo. No entanto, não exploramos todos esses algoritmos, selecionamos alguns deles como resultados de EMANUEL. Para um maior embasamento sobre a estratégia, a Seção 5.1.1 descreve o hiperespaço; a Seção 5.1.2 o algoritmo de otimização multiobjetivo e os objetivos; a Seção 5.1.3 o indivíduo e a Seção 5.1.4 os critérios de seleção de algoritmos da fronteira de Pareto para fins de comparação com os algoritmos *baselines*.

5.1.1 Hiperespaço

EMANUEL avalia diferentes algoritmos e configurações de hiperparâmetros durante a otimização. O hiperespaço especifica quais algoritmos podem ser selecionados e os possíveis valores dos correspondentes hiperparâmetros. O hiperespaço foi definido com base em Sá et al. (2020) e contém 18 algoritmos de classificação monorrótulo e 18 algoritmos de classificação multirrótulo.

Os algoritmos de classificação monorrótulo contidos no hiperespaço são: *Bayesian Network Classifier* (BNC), DT, IBk, J48, JRip, *K Star* (K*), *Logistic Model Trees* (LMT), *Logistic Regression* (LR), MLP, NB, RF, *Random Tree* (RT), *Stochastic Gradient Descent* (SGD), *Simple Logistic* (SL), SMO, *Partial Decision Trees* (PART), REPTree e *Voted Perceptron* (VP).

Os algoritmos de classificação multirrótulo do hiperespaço pertencem às abordagens:

- ❑ **Adaptação de algoritmos**: *Back Propagation Neural Network* (BPNN); e
- ❑ **Transformação de problemas**: BR, *Binary Relevance – Quick Version* (BRq), *Bayesian Classifier Chains* (BCC), CC, CDN, *Conditional Dependency Trellis* (CDT), *Classifier Trellis* (CT), *Four-Class Pairwise Classification* (FW), LP, *Monte-Carlo Classifier Chains* (MCC), *Probabilistic Classifier Chains* (PCC), *Pruned Sets* (PS), *Pruned Sets with Threshold* (PSt), *Ranking and Threshold* (RTh) e *Population of MCC* (PMCC).
- ❑ **Algoritmos ensemble**: RAkEL e RAkELd.

Definimos os hiperparâmetros como um conjunto finito de valores. Embora alguns algoritmos tenham hiperparâmetros contínuos, discretizamos seus valores para conjuntos finitos de números reais porque a representação de hiperparâmetros contínuos exigiria indivíduos muito maiores e isso aumentaria ainda mais a complexidade da estratégia AutoML para classificação multirrótulo. Além disso, a discretização desses hiperparâmetros é um procedimento bem conhecido no AutoML (LE; FU; MOORE, 2020; GIJSBERS; VANSCHOREN, 2021).

O hiperespaço também inclui a possibilidade de normalizar ou não os dados para a classificação, essa opção está vinculada aos algoritmos. Apesar da normalização não ser um hiperparâmetro dos classificadores, mas uma tarefa de pré-processamento, ela foi representada no hiperespaço como um hiperparâmetro condicional, habilitado para algoritmos específicos baseados em distância. Portanto, sua aplicabilidade depende do algoritmo de classificação. Além da normalização, o hiperespaço também modela outras relações condicionais. Para mais informações sobre o hiperespaço, consulte o Apêndice A. O leitor também pode consultar o Material Suplementar de Sá et al. (2020) para obter uma base teórica sobre os algoritmos de hiperespaço.

5.1.2 Otimização Multiobjetivo

EMANUEL utiliza o algoritmo de otimização multiobjetivo NSGA-II. Os objetivos da otimização estão relacionados ao desempenho e à eficiência computacional. Muitos trabalhos em NAS também trabalham nesta linha, limitando a complexidade computacional ou o tamanho do modelo por meio do número de operações de pontos flutuantes (do inglês *floating-point operation*s (FLOPs)) ou do número de parâmetros do modelo (KARL et al., 2023). No entanto, no contexto deste trabalho, obter o número de FLOPs não é uma tarefa simples e não há um parâmetro comum a todos os algoritmos a ser minimizado ou maximizado. O tamanho do modelo foi calculado como o tamanho, em bytes, do arquivo gerado pela serialização do modelo.

Deste modo, os objetivos do problema de otimização multiobjetivo são: i) maximizar o Macro F-score e ii) minimizar o tamanho do modelo. A ideia é que modelos menores são mais simples e eficientes, ao passo que modelos maiores são mais complexos e custosos. Para trabalhar com um problema de minimização no NSGA-II, adotamos minimizar simultaneamente: $-1 * \text{Macro F-score}$, e o tamanho do correspondente modelo. Para obter a aptidão de uma solução candidata, devemos treinar e avaliar o classificador multirrótulo quanto aos objetivos.

5.1.3 Indivíduo

O NSGA-II trabalha com uma população de indivíduos e cada indivíduo é uma codificação de um algoritmo de classificação multirrótulo. A decodificação de um indivíduo em um algoritmo de classificação (solução candidata) é feita a partir do hiperespaço multirrótulo. O indivíduo é representado por um vetor de 28 números inteiros aleatórios. Cada posição do vetor corresponde a um gene: o primeiro gene especifica se os dados serão normalizados ou não; o segundo gene define o algoritmo de classificação multirrótulo e os genes seguintes são os hiperparâmetros do algoritmo.

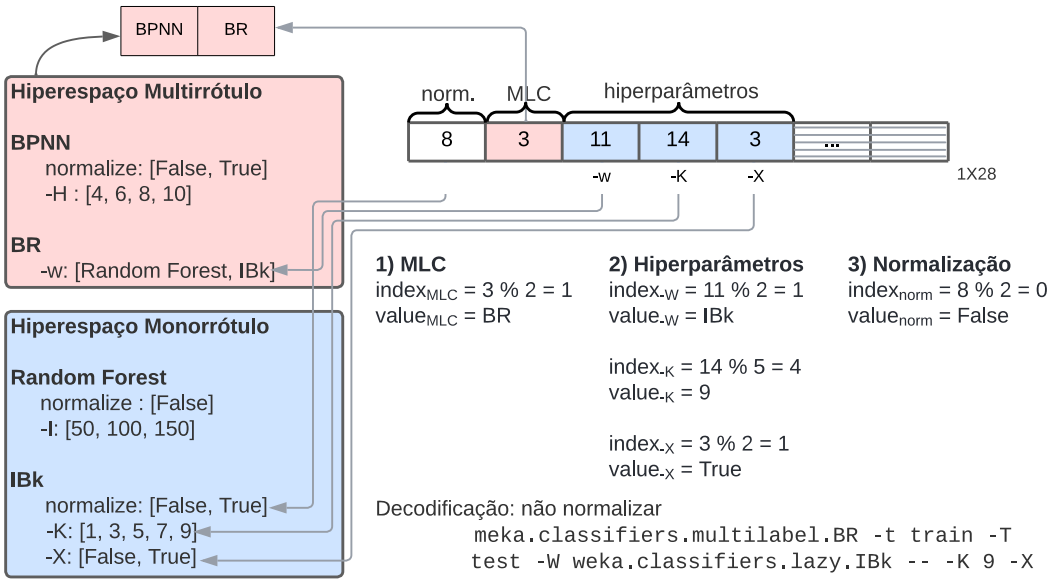
O vetor numérico é decodificado em um algoritmo de classificação multirrótulo durante a otimização. A decodificação deve começar pelo algoritmo de classificação multirrótulo, pois a normalização e os hiperparâmetros dependem do algoritmo de classificação. As Equações 11 e 12 apresentam as fórmulas para a decodificação. Chamamos o valor de um gene do indivíduo de `gene_value` e os vetores do hiperespaço de `array_values`. A Equação 11 encontra um índice calculando o resto da divisão do valor do gene pelo tamanho do vetor de valores do hiperespaço. O índice encontrado refere-se a uma posição de `array_values`. A Equação 12 encontra um valor para o algoritmo de classificação multirrótulo, normalização ou hiperparâmetros, consultando um vetor do hiperespaço em um índice específico.

$$index = gene\_value \% len(array\_values) \tag{11}$$

$$value = array\_values[index] \tag{12}$$

A Figura 14 ilustra um exemplo de decodificação de um indivíduo. O hiperespaço da Figura 14 possui apenas dois algoritmos de classificação multirrótulo para fins didáticos. A decodificação começa com o classificador multirrótulo, é possível obter do hiperespaço um vetor com os algoritmos de classificação multirrótulo. Nesse caso, `array_values = [BPNN, BR]`. O valor do gene para o algoritmo de classificação multirrótulo é três (`gene_value = 3`). Utilizando a Equação 11, o resto da divisão de `gene_value` (três) pelo tamanho de `array_values` (dois) é um, logo `index = 1`. Pela Equação 12, `value` é o algoritmo de classificação multirrótulo `BR` (`value = array_value[1]`).

Figura 14 – Exemplo ilustrativo da representação e decodificação de um indivíduo. Na figura, norm. é a normalização e MLC (*Multi-label Classification*) é o algoritmo de classificação multirrótulo.



A decodificação continua, agora observando os hiperparâmetros do algoritmo `BR` no hiperespaço multirrótulo. Em nosso exemplo, `W` é o único hiperparâmetro, que se refere ao algoritmo de classificação monorrótulo base. Os possíveis valores para `W` são `Random Forest` e `IBk` (`array_values = [Random Forest, IBk]`). Para esse hiperparâmetro, o valor do terceiro gene é 11 (`gene_value = 11`). O resultado da Equação 11 é `index = 11%2 = 1` e da Equação 12 é `value = array_values[1] = IBk`. Portanto, o classificador monorrótulo base é o algoritmo `IBk`.

Seguindo o processo de decodificação, devemos decodificar os hiperparâmetros do algoritmo monorrótulo. No hiperespaço monorrótulo, os hiperparâmetros do algoritmo `IBk` são `normalize`, `K` e `X`. Neste momento, desconsideramos a normalização. Para `K`, há cinco

valores no hiperespaço monorrótulo (`array_values = [1, 3, 5, 7, 9]`), e o valor do quarto gene é 14 (`gene_value = 14`). Assim, o valor de `K` é igual a nove (Equações 11 e 12). Os possíveis valores de `X` no hiperespaço são `False` ou `True`, e o valor do quinto gene é três (`gene_value = 3`). Nesse caso, os resultados das Equações 11 e 12 são um e `True`, respectivamente.

Para finalizar a decodificação, resta determinar o uso ou não da normalização. No primeiro gene está a informação sobre a normalização (`gene_value = 8`) e no hiperespaço há a opção de normalizar ou não os dados (`array_values = [False, True]`). Usando as Equações 11 e 12, o resultado da decodificação é `False`. Portanto, o indivíduo decodificado é o algoritmo `BR`, com classificador base `IBk`, tendo número de vizinhos (`-K`) igual a nove e a opção de selecionar o número de vizinhos entre um e `K` (`-X`) é `True`. Os genes sombreados na Figura 14 possuem valores, mas não são utilizados, estando inativos. Após a decodificação, o *fitness* do indivíduo pode ser calculado.

5.1.4 Seleção de Algoritmos da Fronteira de Pareto

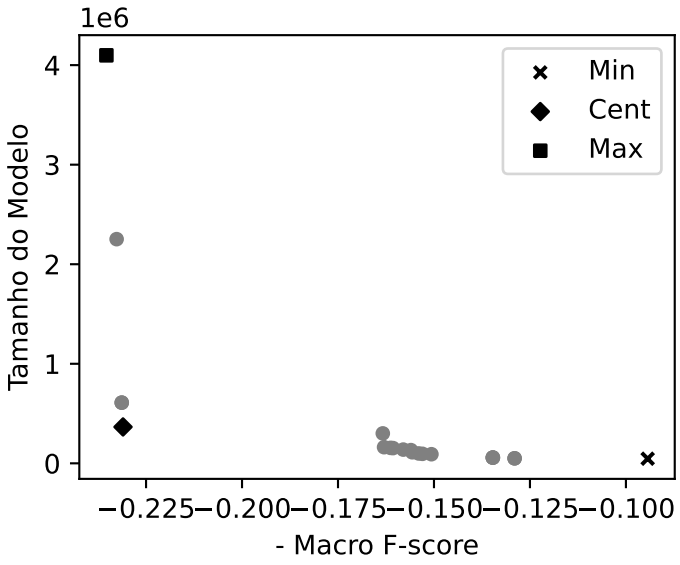
O resultado de EMANUEL é a fronteira de Pareto contendo os algoritmos de classificação multirrótulo que minimizam ambos os valores dos objetivos. Embora seja uma tarefa comum, não há regras que especifiquem como selecionar soluções/algoritmos da fronteira de Pareto. Neste trabalho, os algoritmos selecionados são utilizados para comparar a estratégia proposta com diferentes *baselines*.

Selecionamos três algoritmos multirrótulo dentre os disponíveis na fronteira de Pareto, algoritmos localizados no centro e nos extremos da fronteira, com valores dos objetivos intermediários/centrais, mínimos e máximos. Para escolher a solução central, utilizamos *Frugality Score* (EVCHENKO, 2016), uma métrica que avalia algoritmos do AM combinando uma medida de desempenho e a quantidade de algum recurso. Nesta proposta, *Frugality Score* penaliza Macro F-score com o tamanho do modelo, conforme a Equação 13.

$$frugality_i = macro\_f\_score_i - \frac{0.5}{(1 + 1/tamanho\_modelo\_norm_i)} \tag{13}$$

Frugality Score é calculado para cada algoritmo de classificação multirrótulo da fronteira de Pareto. O algoritmo com *score* mais alto é selecionado como representante médio. Na Equação 13, $macro\_f\_score_i$ corresponde ao valor positivo da medida Macro F-score e $tamanho\_modelo\_norm_i$ ao tamanho do modelo normalizado no intervalo de $[0.01, 1]$ de um algoritmo i. A normalização é necessária, pois os objetivos da otimização possuem escalas diferentes. Na Figura 15, cada ponto do gráfico representa um algoritmo de classificação multirrótulo no espaço dos objetivos e os pontos em destaque representam os algoritmos selecionados da fronteira de Pareto.

Figura 15 – Gráfico da fronteira de Pareto no espaço dos objetivos com três pontos selecionados. Os pontos no formato quadrado, losango e X representam algoritmos com o maior (*Max*), central (*Cent*) e menor (*Min*) Macro F-score e tamanho do modelo, respectivamente.



5.2 Detalhes de Implementação e *Setup* de Execução

Os algoritmos incluídos no hiperespaço (Seção 5.1.1) são implementações disponíveis nas bibliotecas `WEKA` (HALL et al., 2009) e `MEKA` (READ et al., 2016). Como a implementação de EMANUEL foi feita na linguagem de programação `Python`, utilizamos a interface `scikit-multilearn` para `MEKA` para utilizar essas implementações. No entanto, fizemos alterações na implementação da interface para que fosse possível setar todos os hiperparâmetros dos algoritmos.

A estratégia EMANUEL foi implementada com o algoritmo de otimização NSGA-II, com os hiperparâmetros descritos na metodologia experimental (Seção 4.4). A população inicial foi selecionada de forma aleatória, seguindo a representação e a decodificação dos indivíduos em algoritmos de classificação multirrótulo descritos na Seção 5.1.3.

Para contribuir com a execução da estratégia AutoML, EMANUEL limita o tempo de treinamento de um algoritmo de classificação multirrótulo a 30 minutos. Esse limite foi definido, pois a complexidade do algoritmo de classificação multirrótulo e a dimensionalidade dos conjuntos de dados podem tornar o treinamento dos modelos inviável no contexto do AutoML. Caso o tempo de treinamento exceda o limite pré-estabelecido, os objetivos da otimização recebem seus valores máximos (zero para $-1 * \text{Macro F-score}$ e 1 GB para o tamanho do modelo). Além disso, EMANUEL armazena os algoritmos de classificação multirrótulo, suas configurações de hiperparâmetros e os valores dos objetivos após treino/teste para futuras consultas, isso previne que o mesmo classificador seja

treinado repetidas vezes nas gerações do NSGA-II.

Os algoritmos IBLR+, CDN, RAkELo, BR com AdaBoost, CC e ECC foram os algoritmos *baselines* de comparação. A execução dos *baselines* ocorreu nos *3-folds* pré-definidos de um conjunto de dados e foi limitada a sete dias. Tanto EMANUEL quanto os *baselines* foram executados cinco vezes para cada conjunto de dados multirrótulo. Os conjuntos de dados e os algoritmos *baselines* foram descritos no Capítulo 4.

5.3 Comparando EMANUEL com os *Baselines*

Os resultados das execuções de EMANUEL foram cinco fronteiras de Pareto. Utilizando o conceito de não dominância, encontramos a primeira fronteira de Pareto não dominada a partir das cinco fronteiras originais. Isto foi feito para encontrar uma fronteira resultante das fronteiras originais e para apresentar os resultados a partir de uma única fronteira. Selecionamos três algoritmos de classificação multirrótulo da fronteira resultante, conforme descrito na Seção 5.1.4, e nomeamos os algoritmos selecionados com o menor valor, o valor intermediário/central e o maior valor de Macro F-score/tamanho do modelo como EMANUEL$_{Min}$, EMANUEL$_{Cent}$ e EMANUEL$_{Max}$, respectivamente.

Calculamos a média de Macro F-score e do tamanho do modelo a partir dos resultados das cinco execuções de um *baseline* em um conjunto de dados. Os valores médios foram tomados para a discussão dos resultados. Para o conjunto de dados *delicious*, os algoritmos CDN e IBLR+ não concluíram as cinco execuções em tempo inferior a sete dias. Portanto, removemos esse conjunto de dados dos testes estatísticos para evitar viés nas análises.

A Figura 16 apresenta os resultados quanto ao Macro F-score. Os algoritmos de classificação multirrótulo selecionados da fronteira de Pareto resultante de EMANUEL possuem os maiores (EMANUEL$_{Max}$) e os menores (EMANUEL$_{Min}$) Macro F-scores. Os algoritmos selecionados do centro (EMANUEL$_{Cent}$) da fronteira têm Macro F-score inferior a EMANUEL$_{Max}$, mas superior aos *baselines* na maioria dos conjuntos de dados. Dentre os *baselines*, CDN e ECC normalmente apresentam valores altos de Macro F-scores.

Para complementar a análise da Figura 16, a Figura 17 apresenta o *ranking* dos algoritmos avaliados no experimento para todos os conjuntos de dados. Para cada conjunto de dados, os algoritmos foram ordenados pelos *rankings* médios, do menor *ranking* para o maior. Os quadrados azuis na Figura 17 representam os algoritmos com os melhores Macro F-scores e os quadrados vermelhos, os algoritmos com os piores valores para essa medida. A Figura 17 confirma que EMANUEL$_{Max}$, EMANUEL$_{Cent}$, CDN e ECC são os algoritmos com os melhores valores para Macro F-score, enquanto EMANUEL$_{Min}$ possui o pior valor de Macro F-score entre os algoritmos avaliados. Para uma análise numérica do Macro F-score, consulte a Tabela 21 do Apêndice B.

Os resultados para o tamanho do modelo estão na Figura 18. Algoritmos como CDN, ECC e EMANUEL$_{Max}$ são os que possuem os maiores tamanhos de modelos na maioria

Figura 16 – Macro F-scores dos algoritmos selecionados da fronteira de EMANUEL e Macro F-scores médio dos algoritmos *baselines*.

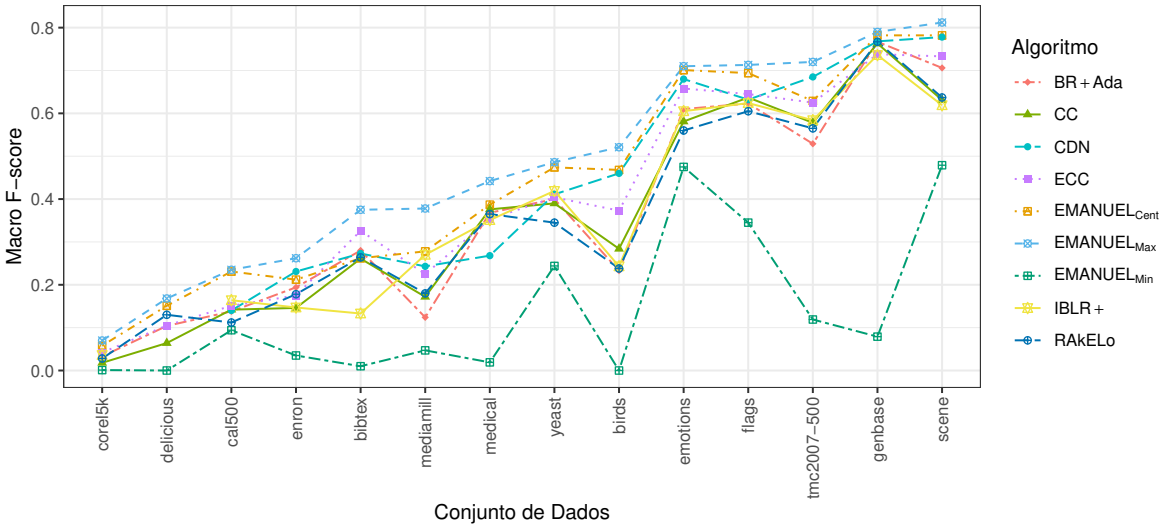
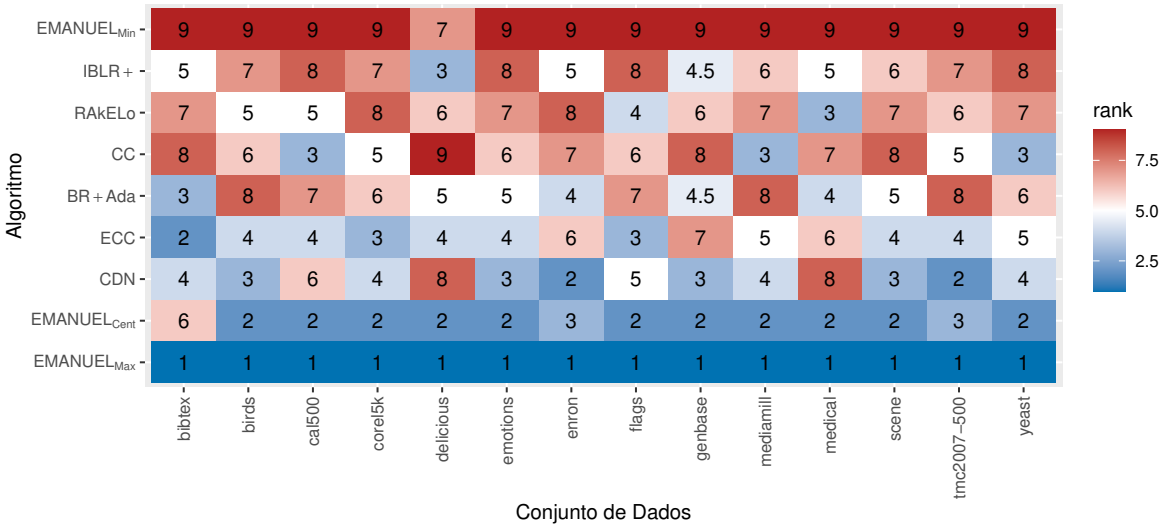


Figura 17 – *Ranking* dos algoritmos avaliados no experimento quanto ao Macro F-score. Na figura, *rankings* iguais para os algoritmos de um conjunto de dados representam algoritmos que tiveram Macro F-scores iguais.



dos conjuntos de dados. EMANUEL$_{Min}$ tem o menor tamanho de modelo em 12 dos 14 conjuntos de dados. Em complemento à Figura 18, a Figura 19 identifica o *ranking* dos algoritmos em relação ao tamanho dos modelos. De acordo com o *ranking*, EMANUEL$_{Min}$, EMANUEL$_{Cent}$ e BR com AdaBoost são os algoritmos que induzem os menores modelos e, EMANUEL$_{Max}$, IBLR+, ECC e CDN são os algoritmos que induzem os maiores modelos. Para uma análise numérica do tamanho dos modelos, consulte a Tabela 22 do Apêndice B.

Outra possível análise é a comparação dos algoritmos selecionados da fronteira de Pareto de EMANUEL com os melhores resultados dos *baselines* para cada conjunto de dados. Nas Figuras 20 e 21, EMANUEL$_{Min}$ é o algoritmo com os menores valores de Macro F-score e do tamanho do modelo. EMANUEL$_{Max}$ é o algoritmo com melhor desempenho,

Figura 18 – Tamanhos dos modelos induzidos pelos algoritmos selecionados da fronteira de EMANUEL e pelos algoritmos *baselines*. Para melhor visualização dos tamanhos dos modelos, utilizou-se o logaritmo dessa medida.

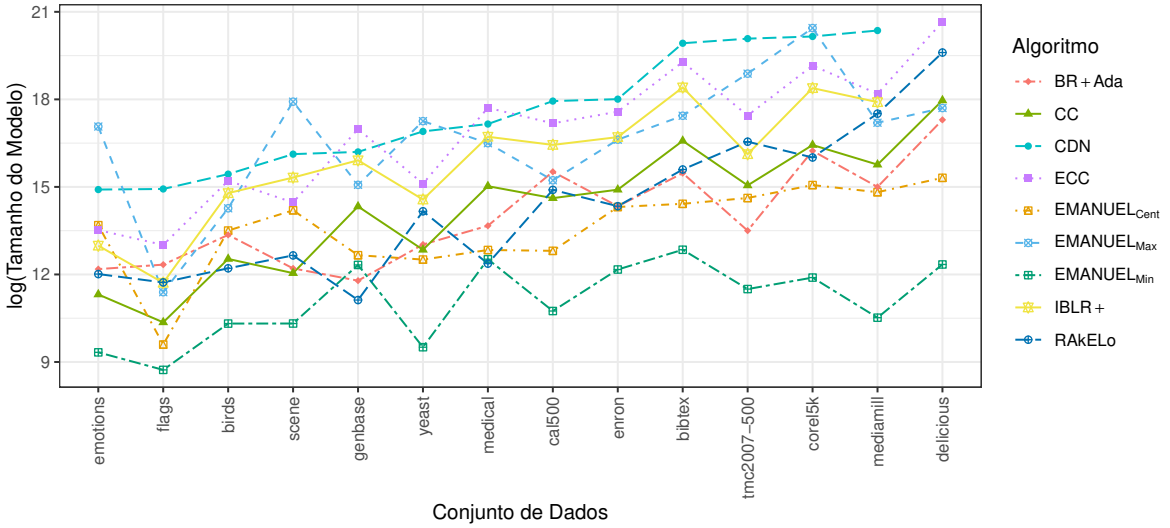
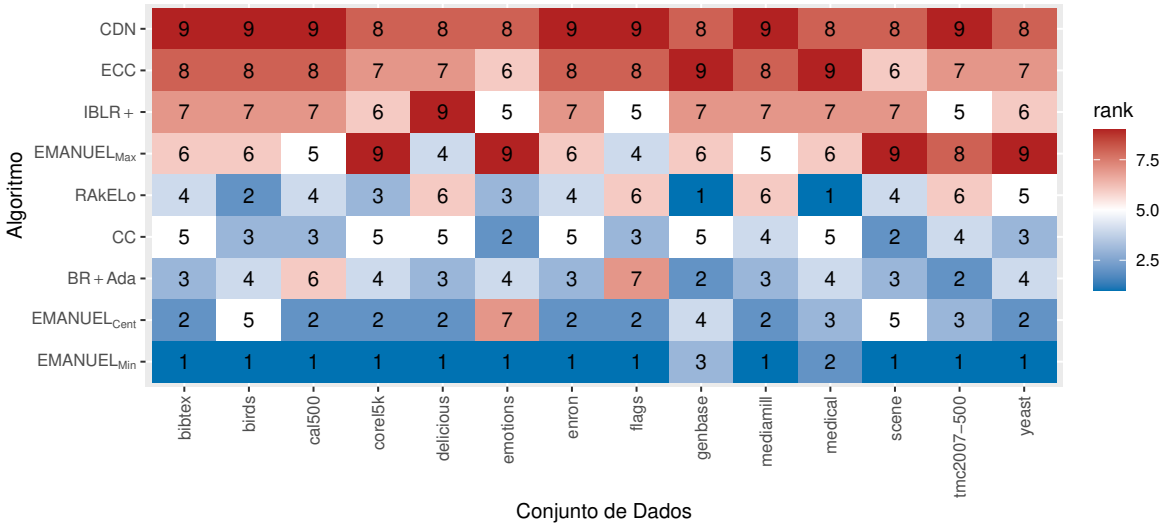


Figura 19 – *Ranking* dos algoritmos avaliados no experimento quanto ao tamanho do modelo. Na figura, *rankings* iguais para os algoritmos de um conjunto de dados representam algoritmos que tiveram tamanho de modelos iguais.



mas ele induz modelos com os maiores tamanhos. EMANUEL$_{Cent}$ e os melhores *baselines* induzem modelos com desempenho e tamanhos intermediários, entre EMANUEL$_{Min}$ e EMANUEL$_{Max}$.

Figura 20 – Macro F-score resultantes de EMANUEL$_{Min}$, EMANUEL$_{Cent}$, EMANUEL$_{Max}$ e dos melhores algoritmos *baselines* de cada conjunto de dados.

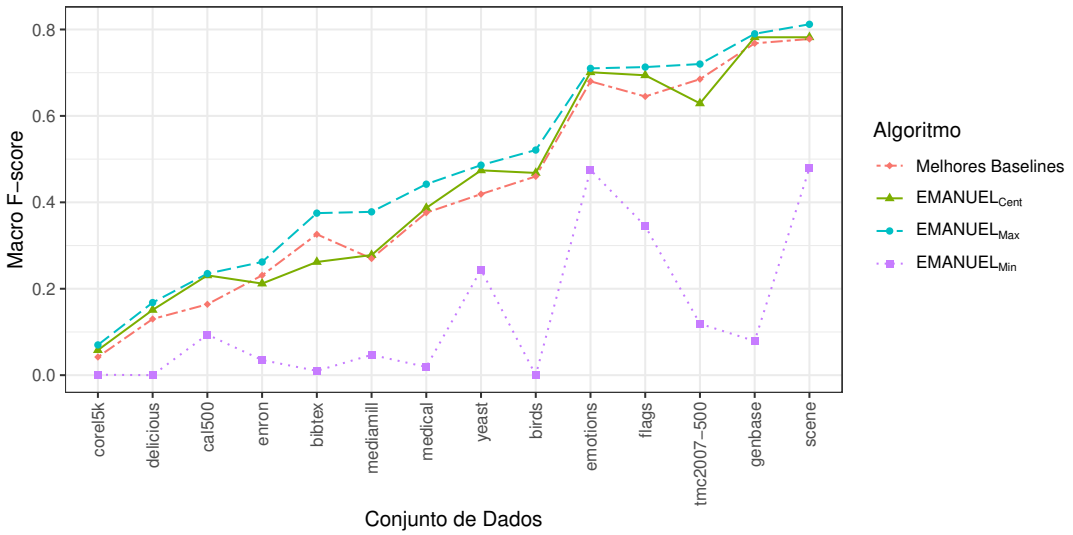
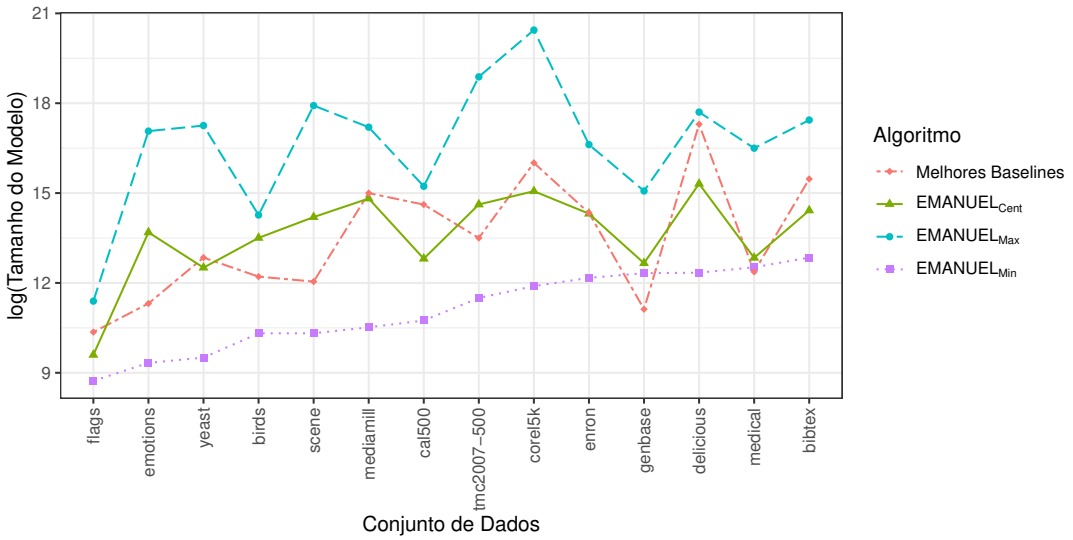


Figura 21 – Tamanho dos modelos induzidos por EMANUEL$_{Min}$, EMANUEL$_{Cent}$, EMANUEL$_{Max}$ e pelos melhores algoritmos *baselines* de cada conjunto de dados.



Para confirmar os resultados discutidos nesta seção, o teste de Friedman foi aplicado para verificar se os algoritmos avaliados possuíam desempenhos semelhantes em relação aos objetivos. Foram feitos dois testes estatísticos: 1) para Macro F-score; e 2) para o tamanho do modelo. A hipótese nula foi rejeitada no primeiro teste, indicando que havia diferenças significativas no Macro F-score dos algoritmos avaliados. O teste *post-hoc* de Nemenyi foi utilizado para encontrar os grupos de algoritmos com Macro F-score diferentes.

Figura 22 – Diagrama de diferença crítica dos algoritmos avaliados em relação ao Macro F-score.

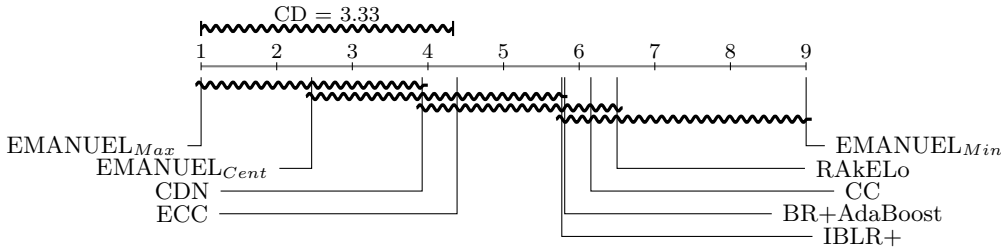
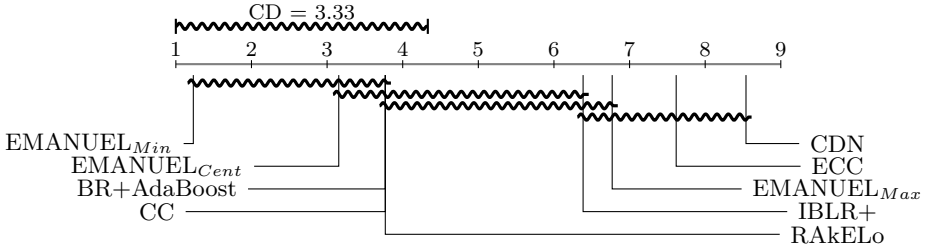


Figura 23 – Diagrama de diferença crítica dos algoritmos avaliados em relação ao tamanho do modelo.



No diagrama de diferença crítica da Figura 22, os algoritmos com melhores classificações (mais à esquerda) possuem os melhores Macro F-scores, enquanto os algoritmos com classificações piores (mais à direita) possuem desempenhos piores. Os algoritmos selecionados das fronteiras de Pareto, EMANUEL$_{Max}$ e EMANUEL$_{Cent}$, bem como o *baseline* CDN foram os algoritmos com os melhores Macro F-scores. Os demais algoritmos apresentaram desempenho inferior (ECC, IBLR+, BR com AdaBoost, CC, RAkELo, e EMANUEL$_{Min}$).

No teste estatístico para o tamanho dos modelos, a hipótese nula também foi rejeitada, ou seja, os modelos possuíam tamanhos diferentes. A partir do teste *post-hoc* de Nemenyi, encontramos os grupos de algoritmos com tamanhos de modelos diferentes. No diagrama de diferença crítica da Figura 23, os algoritmos com melhores *ranks*, ou com os menores tamanhos de modelos, foram os algoritmos EMANUEL$_{Min}$ e EMANUEL$_{Cent}$. Na perspectiva estatística, não há diferenças estatísticas (no que se refere ao tamanho dos modelos) entre os algoritmos supracitados e os algoritmos BR com AdaBoost, CC e RAkELo. Os demais algoritmos tiveram tamanhos de modelos estatisticamente superiores (IBLR+, EMANUEL$_{Max}$, ECC e CDN).

Ao selecionar algoritmos na fronteira de Pareto, se o interesse for otimizar simultaneamente o Macro F-score e o tamanho do modelo, uma boa escolha seria EMANUEL$_{Cent}$, pois esse algoritmo tem bons resultados para ambas as medidas. Tal fato é comprovado pelos testes estatísticos, na Figura 22, EMANUEL$_{Cent}$ está entre os algoritmos com me-

lhores Macro F-scores e, na Figura 23, EMANUEL$_{Cent}$ está entre os algoritmos com os menores tamanhos de modelos. Seguindo essa análise, algoritmos como EMANUEL$_{Max}$, CDN e ECC possuem desempenho e tamanho de modelos altos. Em contrapartida, algoritmos como EMANUEL$_{Min}$, BR com AdaBoost e CC possuem desempenho e tamanho de modelos baixos.

Dessa forma, os resultados experimentais confirmaram a hipótese de pesquisa. A estratégia EMANUEL é capaz de encontrar uma fronteira de Pareto diversificada com classificadores multirrótulo que representam diferentes compensações entre os objetivos da otimização. Os classificadores da fronteira podem ter desempenho competitivo em relação aos *baselines* a depender do classificador selecionado. Classificadores selecionados na parte central da fronteira tendem a ser competitivos para ambos os objetivos. EMANUEL$_{Cent}$, por exemplo, é um algoritmo que tem bons resultados tanto para o Macro F-score quanto para o tamanho dos modelos. Comparando EMANUEL$_{Cent}$ com o algoritmo *baseline* com o melhor Macro F-score (CDN), ambos apresentam equivalência estatística para essa medida. No entanto, EMANUEL$_{Cent}$ destaca-se por induzir modelos de tamanhos estatisticamente inferiores.

Já os classificadores selecionados nos extremos da fronteira (*Min* e *Max*) tendem a otimizar um dos objetivos. EMANUEL$_{Max}$ obteve o maior Macro F-score dentre os algoritmos avaliados e está entre os algoritmos com os piores resultados para o tamanho dos modelos. De forma similar, EMANUEL$_{Min}$ induz os menores modelos e tem o pior resultado para o Macro F-score. Esse classificador apresenta resultados limitados, mas será mantido nas discussões dos resultados para representar as escolhas nas fronteiras. Logo, EMANUEL alcançou resultados competitivos considerando os três classificadores (*Min*, *Cent* e *Max*) que representaram a estratégia.

5.4 Considerações Finais

Neste capítulo apresentamos EMANUEL, uma estratégia AutoML multiobjetivo para classificação multirrótulo que encontra um conjunto de modelos com diferentes desempenhos em relação à medida Macro F-score e ao tamanho dos modelos. A estratégia EMANUEL foi desenvolvida com NSGA-II, onde os indivíduos da população representam algoritmos de classificação multirrótulo. A aptidão de um indivíduo foi obtida por meio da indução e da avaliação dos modelos de classificação multirrótulo.

Para verificar o desempenho da estratégia quanto aos objetivos, três algoritmos foram selecionados da fronteira de Pareto resultante de EMANUEL e comparados com os algoritmos *baselines*. EMANUEL$_{Max}$, EMANUEL$_{Cent}$ e CDN tiveram um desempenho estatisticamente melhor que os demais algoritmos avaliados em relação ao Macro F-score. EMANUEL$_{Min}$, EMANUEL$_{Cent}$, BR com AdaBoost e CC tiveram tamanho do modelo estatisticamente menor que os demais algoritmos.

As soluções de EMANUEL foram estatisticamente superiores aos demais algoritmos, estando no grupo dos melhores algoritmos tanto para Macro F-score quanto para o tamanho do modelo. A solução EMANUEL$_{Cent}$ foi a única solução dentre as avaliadas que conseguiu resultados estatisticamente melhores para ambos os objetivos. Isso demonstra que a estratégia proposta encontra automaticamente soluções que ponderam diferentes objetivos. No entanto, o tempo de execução de EMANUEL é elevado. Na tentativa de melhorar o tempo de execução de EMANUEL, o próximo capítulo explora o uso de modelos substitutos. Assim, todos os dados gerados a partir da execução de EMANUEL constituem meta-conhecimento que será utilizado pelo modelo substituto proposto no Capítulo 6.

Capítulo 6

EMANUEL$_{SM}$

O AutoML automatiza a seleção de algoritmos e seus hiperparâmetros para uma tarefa. Uma forma convencional de automatização é testar possíveis soluções em busca daquela que otimiza o(s) objetivo(s). Isso envolve a indução e o teste de modelos de AM até que uma solução seja encontrada, o que torna o AutoML computacionalmente custoso. Uma alternativa recente para este problema é o uso do meta-aprendizado (BRAZDIL et al., 2022). Segundo Vanschoren (2019), o meta-aprendizado é o aprendizado a partir das experiências anteriores em uma tarefa. Em outras palavras, o meta-aprendizado utiliza o conhecimento acumulado em tarefas anteriores para resolver uma nova tarefa.

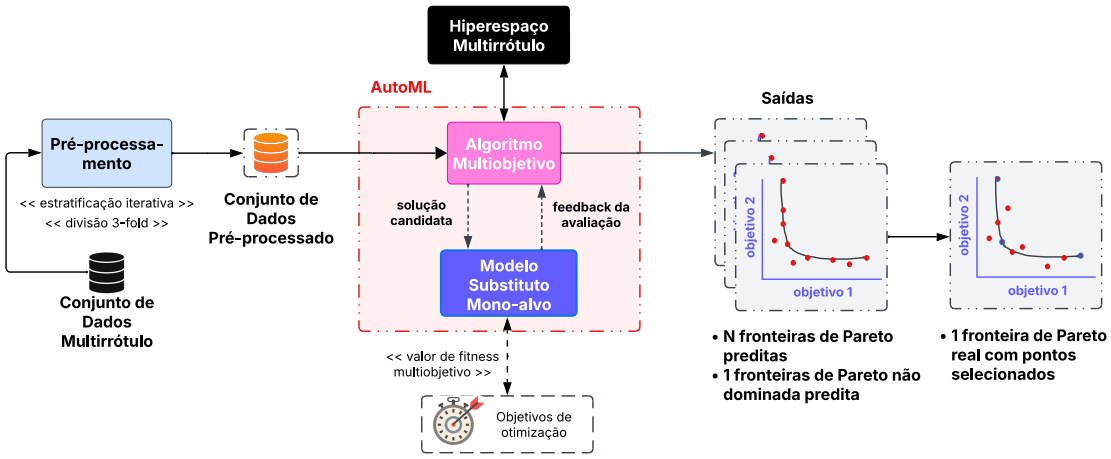
Este capítulo investiga o uso do meta-aprendizado em uma estratégia AutoML multi-objetivo para classificação multirrótulo por meio de modelos substitutos. Esses modelos são responsáveis por estimar o desempenho dos algoritmos de classificação multirrótulo durante a otimização, reduzindo o custo computacional do AutoML. Para isso, incluímos os modelos substitutos em EMANUEL e denominamos a nova estratégia de *gEnetic Multi-objective strategy for the Automatic selectioN of mUlti-labEl cLassifiers based on Surrogate Models* (EMANUEL$_{SM}$). Nossa hipótese de pesquisa é que EMANUEL$_{SM}$ pode manter a qualidade preditiva dos modelos finais, reduzindo o tempo total de execução do AutoML.

As próximas seções deste capítulo estão organizadas da seguinte forma: a Seção 6.1 descreve o funcionamento de EMANUEL$_{SM}$; a Seção 6.2 apresenta a criação dos metadados utilizados na indução dos modelos substitutos; a Seção 6.3 descreve o processo de indução dos modelos substitutos; a Seção 6.4 detalha a implementação e o *setup* de execução; a Seção 6.5 apresenta os *baselines* de comparação; a Seção 6.6 apresenta e discute os resultados; e a Seção 6.7 faz as considerações finais.

6.1 Como EMANUEL$_{SM}$ Trabalha?

O funcionamento de EMANUEL$_{SM}$ é semelhante ao funcionamento de EMANUEL. A exceção está no papel do avaliador e na forma como as fronteiras de Pareto resultantes são tratadas. Em EMANUEL, o avaliador é responsável por treinar e testar os classificadores multirrótulo para calcular os valores dos objetivos (Figura 13). Esse papel é modificado em EMANUEL$_{SM}$. Conforme ilustra a Figura 24, o avaliador é substituído por modelos substitutos que preveem os valores dos objetivos quando diferentes algoritmos de classificação multirrótulo/indivíduos são consultados. A vantagem é que os modelos substitutos avaliam os algoritmos de classificação sem ter que induzir o modelo em si, o que reduz o custo computacional da estratégia como um todo.

Figura 24 – Fluxograma do funcionamento da estratégia EMANUEL$_{SM}$.



O resultado de EMANUEL$_{SM}$ é uma fronteira de Pareto cujos valores dos objetivos são valores previstos. Assim como em EMANUEL, para N execuções de EMANUEL$_{SM}$ temos N fronteiras de Pareto e utilizamos o conceito de não dominância para encontrar a primeira fronteira de Pareto não dominada (DEB et al., 2002). No entanto, essa fronteira contém valores previstos. Para trabalhar com fronteiras de Pareto reais, induzimos os algoritmos correspondentes da fronteira prevista e, em seguida, recalculamos a primeira fronteira não dominada. A partir dessa fronteira, selecionamos alguns algoritmos como resultados de EMANUEL$_{SM}$. Neste experimento, o algoritmo de otimização adotado foi o NSGA-II e os objetivos da otimização foram o Macro F-score e o tamanho do modelo.

6.2 Criando os Metadados

Para treinar modelos substitutos precisos são necessários metadados contendo o histórico das execuções anteriores dos algoritmos de classificação multirrótulo nos conjuntos

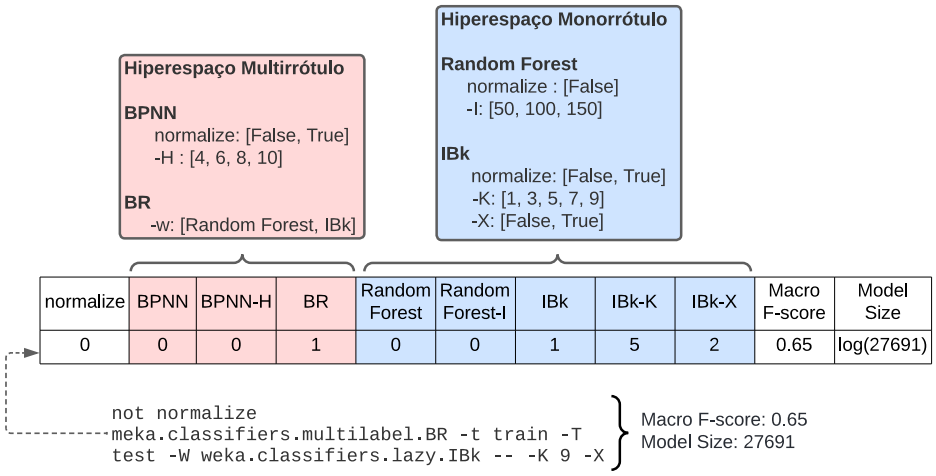
de dados, incluindo o algoritmo de classificação, a configuração de hiperparâmetros e as medidas de desempenho obtidas. Cada algoritmo de classificação multirrótulo foi avaliado nos *3-folds* pré-definidos. Os meta-atributos alvo dos metadados foram agregados como o valor mediano do Macro F-score e do tamanho do modelo resultantes dos *3-folds*. Considerando a geração de metadados, exploramos duas alternativas distintas de meta-conjunto de dados (do inglês *meta-datasets*):

- **Meta-conjunto de dados baseado na estratégia AutoML ($\mathbf{MtDt_A}$):** contém metadados referentes aos algoritmos de classificação multirrótulo treinados e testados com EMANUEL (Capítulo 5). O meta-conjunto de dados resultante possui majoritariamente os algoritmos de classificação que mais minimizam os valores dos objetivos e mais participam do processo evolutivo do NSGA-II.
- **Meta-conjunto de dados baseado no algoritmo Random Search ($\mathbf{MtDt_R}$):** contém metadados resultantes da execução do algoritmo MORS (Seção 2.3.3.1), que seleciona algoritmos de classificação multirrótulo do hiperespaço de forma aleatória, treina e testa o classificador. Esse meta-conjunto de dados contém algoritmos de classificação gerais, com diferentes valores para Macro F-score e para o tamanho do modelo.

Para criar um meta-conjunto de dados, cada algoritmo e hiperparâmetro no hiperespaço corresponde a um meta-atributo de entrada. O primeiro meta-atributo de entrada está relacionado à normalização. Os meta-atributos subsequentes estão relacionados aos algoritmos de classificação multirrótulo e monorrótulo, bem como aos seus hiperparâmetros. Eles aparecem no meta-conjunto de dados na mesma ordem em que foram definidos no hiperespaço. Os valores dos meta-atributos relacionados à normalização e aos algoritmos de classificação monorrótulo/multirrótulo são binários, indicam se houve ou não normalização e definem qual algoritmo de classificação a instância do meta-conjunto de dados representa. Já os valores relacionados aos hiperparâmetros são números inteiros, representando índices dos vetores (iniciando em um) de hiperparâmetros definidos no hiperespaço.

Para exemplificar como os meta-conjuntos de dados são criados, a Figura 25 ilustra um exemplo de um meta-conjunto de dados para um hiperespaço reduzido. Para o hiperespaço da Figura 25, os meta-atributos de entrada são: `normalize`, `BPNN`, `BPNN-H`, `BR`, `Random Forest`, `Random Forest-I`, `IBk`, `IBk-k` e `IBk-X`. Nesse exemplo, o hiperparâmetro `-W` não é considerado um meta-atributo porque corresponde a um algoritmo de classificação monorrótulo (esse meta-atributo já foi incluído). Os meta-atributos alvo são relacionados às medidas Macro F-score e ao tamanho do modelo. A Figura 25 também traz um algoritmo de classificação multirrótulo e os resultados de Macro F-score e do tamanho do modelo induzido por esse algoritmo. A codificação do algoritmo de classificação multirrótulo da Figura 25 representa uma instância do meta-conjunto de dados.

Figura 25 – Exemplo de um meta-conjunto de dados para um hiperespaço reduzido.



O hiperespaço de EMANUEL$_{SM}$ é igual ao hiperespaço de EMANUEL (descrito na Seção 5.1.1 e detalhado no Apêndice A). O meta-conjunto de dados criado a partir desse hiperespaço possui 157 meta-atributos de entrada e dois meta-atributos alvo relacionados aos objetivos da otimização multiobjetivo. Os valores de Macro F-score estão sempre no intervalo de 0 e 1 e os valores dos tamanhos dos modelos não estão limitados a um intervalo e podem ser muito discrepantes. Portanto, para reduzir o intervalo do tamanho do modelo a valores menores, o alvo preditivo adotado foi o logaritmo (log) do tamanho do modelo, em vez de seu valor original. Com os meta-conjuntos de dados criados, foi possível treinar um modelo substituto para cada conjunto de dados.

6.3 Criando Modelos Substitutos

Os modelos substitutos foram induzidos usando o algoritmo de regressão *Random Forest* (RF). RF é um dos principais algoritmos relatados na literatura para treinar modelos substitutos devido à sua habilidade de lidar com espaços de configuração condicionais, parcialmente discretos e de alta dimensão (HOOF; VANSCHOREN, 2021). Os modelos substitutos foram induzidos usando o algoritmo de regressão RF da biblioteca `scikit-learn` (PEDREGOSA et al., 2011) com hiperparâmetros padrão ($n\_estimators = 100$), pois RF normalmente fornece bons resultados com esta configuração (FERNÁNDEZ-DELGADO et al., 2014) para uma ampla gama de aplicações e tarefas de diferentes domínios.

Para definir se os modelos substitutos seriam mono-alvo ou multi-alvo e qual seria o melhor meta-conjunto de dados para indução desses modelos, conduzimos dois experimentos para identificar a melhor configuração para os modelos substitutos em termos de R^2. O primeiro experimento avaliou a versão do modelo a ser treinado. Para isso, repetiu-

se treino/teste de RF dez vezes usando validação cruzada *10-fold*, no meta-conjunto de dados resultante da união de MtDt_A e MtDt_R (MtDt_{A+R}). Os resultados foram avaliados estatisticamente pelo teste não paramétrico de Wilcoxon ($alpha = 5\%$) e sugeriram que os modelos substitutos mono-alvo são os mais precisos. Assim, treinamos dois modelos substitutos, um para prever o Macro F-score e outro para prever o logaritmo do tamanho do modelo.

O segundo experimento avaliou o melhor meta-conjunto de dados para a indução dos modelos substitutos (MtDt_A, MtDt_R ou MtDt_{A+R}). Modelos RF foram induzidos dez vezes com validação cruzada *10-fold* nos diferentes meta-conjuntos de dados. Os resultados foram avaliados estatisticamente pelo teste de Friedman ($alpha = 5\%$) e indicaram que MtDt_{A+R} produz modelos mais precisos em relação à R^2. Assim, MtDt_{A+R} foi o meta-conjunto de dados utilizado para a indução dos modelos substitutos. Para mais informações sobre os experimentos supracitados e seus resultados, consulte o Apêndice C.

6.4 Detalhes de Implementação e *Setup* de Execução

A estratégia EMANUEL_{SM} também foi implementada na linguagem de programação `Python`, nos moldes de EMANUEL, com hiperespaço (Seção 5.1.1), representação de indivíduos (Seção 5.1.3), algoritmo e objetivos de otimização (Seção 5.1.2), operadores de mutação e de *crossover*, tamanho de população, número de gerações e critérios para parada antecipada iguais aos de EMANUEL. No entanto, EMANUEL_{SM} inclui os modelos substitutos para prever os valores dos objetivos. Em consequência disso, esta estratégia não armazena os algoritmos de classificação multirrótulo já avaliados para futuras consultas e não trabalha com um tempo limite para o treinamento dos classificadores como ocorre em EMANUEL.

Os indivíduos da população são avaliados pelo modelo substituto, que prevê o desempenho de um algoritmo de classificação multirrótulo. Os valores previstos são a mediana de Macro F-score e do logaritmo do tamanho do modelo, uma vez que os modelos substitutos foram induzidos utilizando os valores medianos destas métricas nos *3-folds* de um conjunto de dados. Assim, EMANUEL trabalha com valores médios dos objetivos de otimização, enquanto EMANUEL_{SM} utiliza valores medianos. Para otimização com NSGA-II, o tamanho do modelo encontrado pelo modelo substituto foi calculado a partir dos valores do logaritmo desta medida.

As estratégias EMANUEL e EMANUEL_{SM} foram executadas cinco vezes para cada conjunto de dados multirrótulo (descritos na Seção 4.1). As fronteiras de Pareto resultantes de ambas as estratégias foram avaliadas por meio do hipervolume (Seção 2.3.4), onde o ponto de referência foi 0 para $-1 *$ Macro F-score e 1 GB para o tamanho do modelo. Também selecionamos três algoritmos dessas fronteiras, conforme ocorre em EMANUEL (Seção 5.1.4), para comparações com algoritmos *baselines*.

6.5 *Baselines* de Comparação

Comparamos os três algoritmos selecionados das fronteiras resultantes de EMANUEL$_{SM}$ e de EMANUEL com algoritmos de classificação multirrótulo e outras estratégias AutoML. A execução dos algoritmos *baselines* ocorreu nos *3-folds* pré-definidos de um conjunto de dados. IBLR+, CDN, RAkELo, BR com AdaBoost, CC e ECC foram os algoritmos multirrótulo. Cada algoritmo multirrótulo foi executado cinco vezes, com limite de sete dias para cada execução.

Auto-MEKA e `auto-sklearn` foram os *baselines* AutoML. Para `auto-sklearn`, configuramos a automatização da tarefa *pipeline* relacionada à busca dos modelos do AM. Limitamos o tempo de execução de um algoritmo de classificação multirrótulo a 30 minutos. O tempo total de execução de `auto-sklearn` para um conjunto de dados foi o tempo médio gasto por EMANUEL nesse conjunto. Para conjuntos de dados maiores, em que `auto-sklearn` não terminou no tempo total definido, reexecutamos `auto-sklearn` com tempo total de execução de um dia. Um dia é o limite de tempo total sugerido no manual do `auto-sklearn` (FEURER et al., 2015). Para cada conjunto de dados, repetimos a execução de `auto-sklearn` cinco vezes.

Em `Auto-MEKA` o tempo de execução de um algoritmo de classificação multirrótulo também foi de 30 minutos e o tempo total de execução de `Auto-MEKA` para um conjunto de dados foi limitado ao tempo médio gasto por EMANUEL no mesmo conjunto de dados. A execução de `Auto-MEKA` ocorreu por fold de um conjunto de dados. Após a execução nos *3-folds*, repetimos o treinamento e o teste dos melhores algoritmos encontrados por `Auto-MEKA` para calcular os valores médios de Macro F-score e do tamanho do modelo. No entanto, a execução por fold e o retreinamento dos algoritmos de classificação multirrótulo dificultaram a execução de `Auto-MEKA`. Assim, executamos `Auto-MEKA` uma vez nos três folds, isso equivale a uma execução de EMANUEL e de `auto-sklearn`. Os conjuntos de dados e os algoritmos *baselines* utilizados neste experimento estão descritos no capítulo de metodologia experimental (Capítulo 4).

6.6 Resultados

A partir das cinco execuções das estratégias, encontramos a primeira fronteira de Pareto não dominada. Esta fronteira resultante foi a base para a discussão dos resultados. No entanto, em EMANUEL$_{SM}$, a primeira fronteira de Pareto não dominada continha os valores previstos pelos modelos substitutos. Para trabalhar apenas com valores reais, foi necessário treinar e testar os algoritmos de classificação multirrótulo da fronteira de Pareto prevista usando os *3-folds* pré-definidos. Valores médios e medianos resultantes das três partições foram calculados para cada objetivo. Com os valores medianos, foi possível verificar o desempenho dos modelos substitutos e, com os valores médios, recalculou-se a

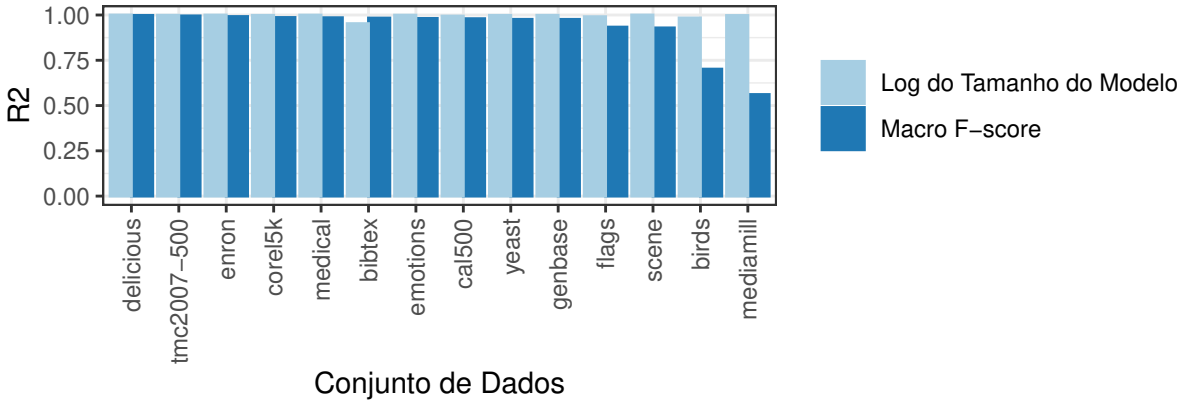
primeira fronteira de Pareto não dominada para possíveis comparações com EMANUEL.

As próximas subseções desta seção estão organizadas da seguinte forma: a Seção 6.6.1 discute o desempenho dos modelos substitutos; a Seção 6.6.2 traça um paralelo entre os resultados de EMANUEL e EMANUEL$_{SM}$; e a Seção 6.6.3 apresenta uma comparação do desempenho das estratégias propostas e dos algoritmos *baselines*.

6.6.1 Desempenho dos Modelos Substitutos

Para avaliar os modelos substitutos, utilizamos a fronteira de Pareto prevista. Para cada algoritmo dessa fronteira, conhecemos o valor mediano real e o valor mediano previsto para cada objetivo. Com esses valores, foi possível calcular o R^2 dos modelos substitutos. O gráfico de barras da Figura 26 mostra os valores de R^2 dos modelos substitutos para os conjuntos de dados investigados. Quanto mais próximo de um estiver o valor de R^2, mais o modelo se ajusta aos dados.

Figura 26 – R^2 dos modelos substitutos que preveem Macro F-score e o logaritmo do tamanho do modelo.



Na Figura 26, os modelos substitutos que preveem o logaritmo do tamanho do modelo tiveram R^2 superior, acima de 0.95. Salvo a exceção dos conjuntos de dados *birds* e *mediamill*, os modelos substitutos que preveem Macro F-score tiveram R^2 acima de 0.92. Nos conjuntos de dados *birds* e *mediamill* os valores de R^2 foram aproximadamente 0.70 e 0.56, respectivamente. Esses valores inferiores para R^2 indicam que os modelos substitutos desses conjuntos de dados tiveram uma taxa de erro maior ao prever o valor de Macro F-score para alguns algoritmos de classificação multirrótulo.

A principal causa dos erros foram os metadados que não representaram totalmente os algoritmos de classificação multirrótulo, bem como seus hiperparâmetros, para os conjuntos de dados em questão. Para confirmar esta informação, plotamos gráficos de dispersão com os valores reais e previstos de Macro F-score para os conjuntos de dados *mediamill*

e *tmc2007-500* (Figuras 27 e 28), que apresentaram os piores e melhores valores de R^2 entre os conjuntos de dados avaliados.

Figura 27 – Diagrama de dispersão para o Macro F-score do conjunto de dados de *mediamill*.

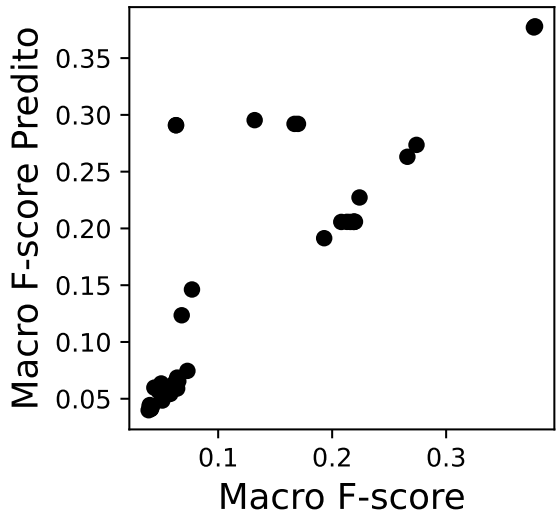
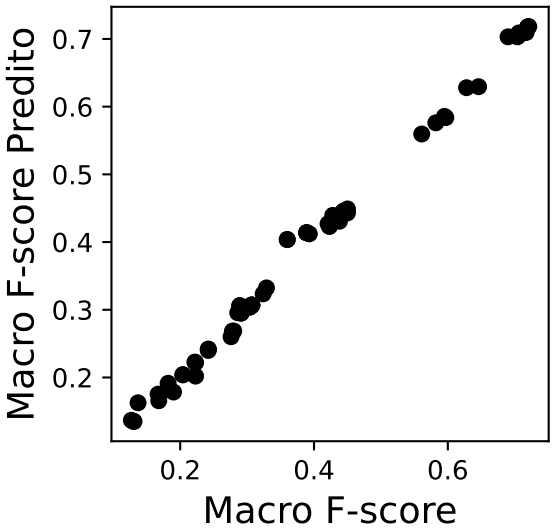


Figura 28 – Diagrama de dispersão para o Macro F-score do conjunto de dados de *tmc2007-500*.



O eixo x do diagrama de dispersão representa os valores reais, enquanto o eixo y representa os valores previstos. Idealmente, se as previsões forem perfeitas, os pontos ficam ao longo de uma linha reta com inclinação um. O gráfico da Figura 27 apresenta o diagrama de dispersão do Macro F-score para o conjunto de dados *mediamill*. Os pontos do diagrama formam uma linha inclinada, mas com dispersão acima da linha, reforçando que o modelo substituto faz previsões corretas ou com erros pequenos, mas também prevê valores acima do esperado. A Figura 28 apresenta o diagrama de dispersão para o conjunto de dados *tmc2007-500*. Neste caso, os pontos do diagrama formam a linha inclinada com pouca dispersão, indicando que as previsões do modelo substituto são geralmente precisas ou próximas das previsões reais.

6.6.2 EMANUEL x EMANUEL$_{SM}$

Os gráficos das Figuras 26, 27 e 28 reforçam a necessidade de utilizar fronteiras de Pareto reais em vez das fronteiras previstas por EMANUEL$_{SM}$, pois os modelos substitutos podem errar suas previsões. Conforme já mencionado, com os valores reais médios de Macro F-score e do tamanho do modelo dos algoritmos de classificação multirrótulo da fronteira prevista, calculamos a fronteira de Pareto real de EMANUEL$_{SM}$. Assim, podemos traçar um paralelo entre os resultados de EMANUEL e de EMANUEL$_{SM}$.

Começamos analisando os algoritmos de classificação multirrótulo das fronteiras. As fronteiras de EMANUEL tiveram maior número e diversidade de algoritmos de classificação multirrótulo do que as fronteiras de EMANUEL$_{SM}$. No entanto, ambas as fronteiras

tinham algoritmos semelhantes, exceto pela configuração dos hiperparâmetros. Além das questões aleatórias que influenciam nos resultados de ambas as estratégias, o que determinou o número de algoritmos na fronteira de EMANUEL$_{SM}$ foi o cálculo da primeira fronteira não dominada considerando os valores reais dos objetivos – muitos pontos (algoritmos de classificação multirrótulo) que eram não dominados passaram a ser dominados por outros pontos.

Para exemplificar, as Figuras 29 e 30 identificam os algoritmos de classificação multirrótulo e monorrótulo das fronteiras de Pareto resultantes das estratégias para o conjunto de dados *medical*. A fronteira resultante de EMANUEL contém 92 algoritmos de classificação multirrótulo com diferentes configurações de hiperparâmetros. Em contraste, a fronteira real resultante de EMANUEL$_{SM}$ contém 36 algoritmos. A maioria dos algoritmos na fronteira de EMANUEL$_{SM}$ (Figura 30) está na fronteira EMANUEL (Figura 29). As fronteiras resultantes das estratégias em outros conjuntos de dados apresentam as mesmas características.

Figura 29 – Heatmap dos algoritmos da fronteira de Pareto de EMANUEL para o conjunto de dados *medical*.

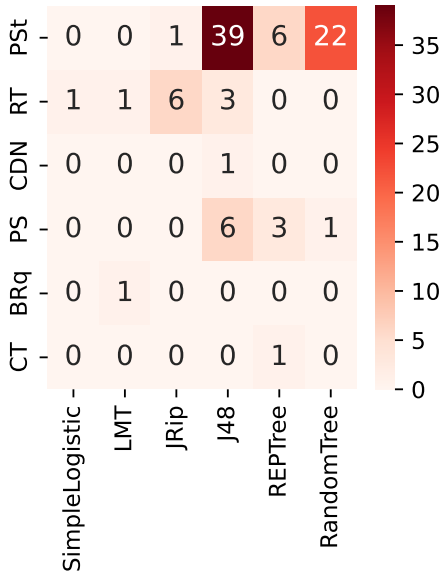
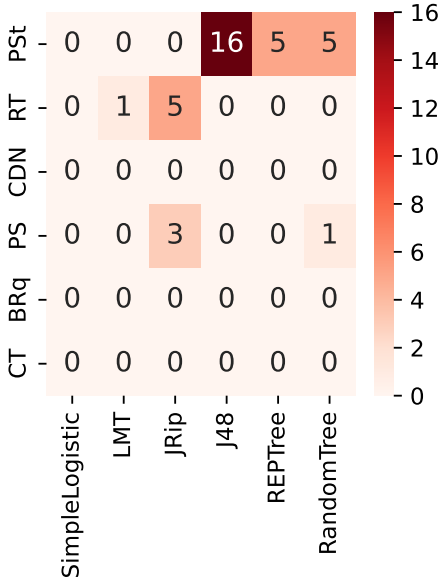
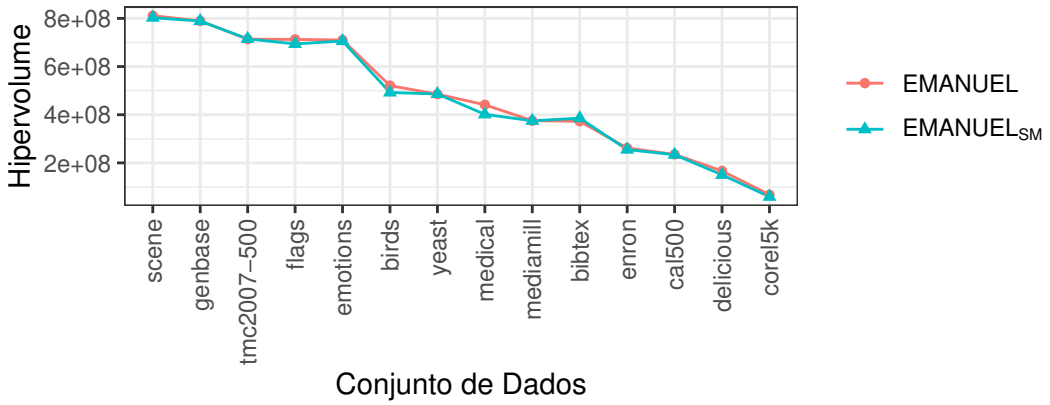


Figura 30 – Heatmap dos algoritmos da fronteira de Pareto de EMANUEL$_{SM}$ para o conjunto de dados *medical*.



Podemos avaliar as fronteiras de Pareto usando o hipervolume. A Figura 31 apresenta os hipervolumes das fronteiras das estratégias propostas. Os valores de hipervolumes foram similares nos conjuntos de dados avaliados. No entanto, o hipervolume resultante da estratégia EMANUEL foi ligeiramente superior ao hipervolume resultante de EMANUEL$_{SM}$ na maioria dos conjuntos de dados. As maiores diferenças de hipervolumes ocorreram nos conjuntos de dados *birds*, *flags* e *medical*. Valores maiores de hipervolumes representam fronteiras que minimizam mais os valores dos objetivos. Isso

Figura 31 – Hipervolume das fronteiras de Pareto de EMANUEL e de EMANUEL$_{SM}$.



significa que EMANUEL encontra resultados que otimizam mais os valores dos objetivos do que EMANUEL$_{SM}$.

Em complemento à análise do hipervolume, avaliamos os três algoritmos de classificação multirrótulo selecionados de ambas as fronteiras quanto ao Macro F-score e ao tamanho do modelo. As Figuras 32 e 33 ilustram os resultados para o Macro F-score e para o tamanho do modelo, respectivamente. Em EMANUEL, nomeamos os algoritmos selecionados com menor valor, valor intermediário/central e maior valor de Macro F-score/tamanho do modelo como EMANUEL$_{\text{Min}}$, EMANUEL$_{\text{Cent}}$ e EMANUEL$_{\text{Max}}$, respectivamente. De forma similar, nomeamos os algoritmos de EMANUEL$_{SM}$ como EMANUEL$_{\text{SM}_{\text{Min}}}$, EMANUEL$_{\text{SM}_{\text{Cent}}}$ e EMANUEL$_{\text{SM}_{\text{Max}}}$, respectivamente.

Figura 32 – Macro F-scores dos modelos induzidos pelos algoritmos selecionados das fronteiras de EMANUEL e EMANUEL$_{SM}$.

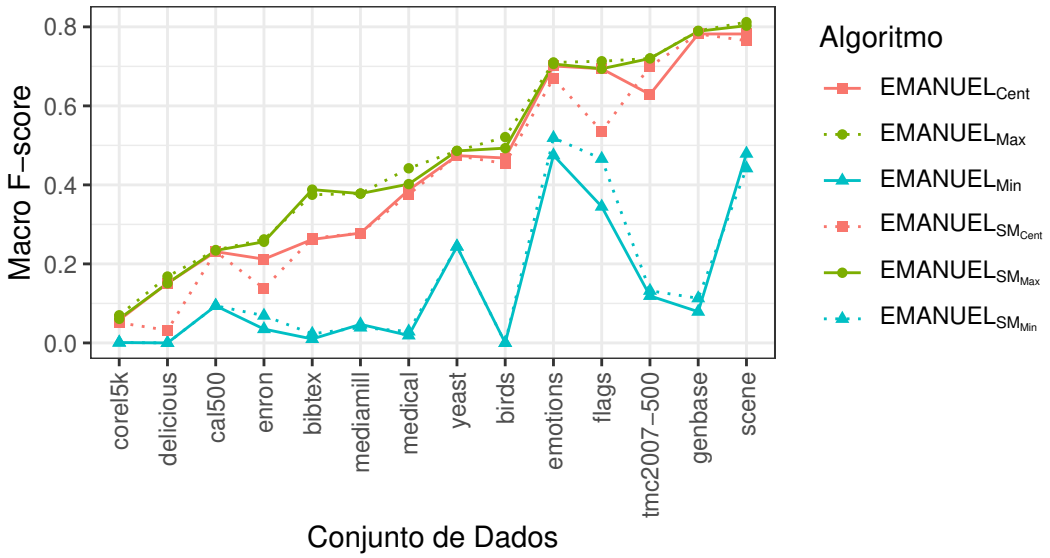
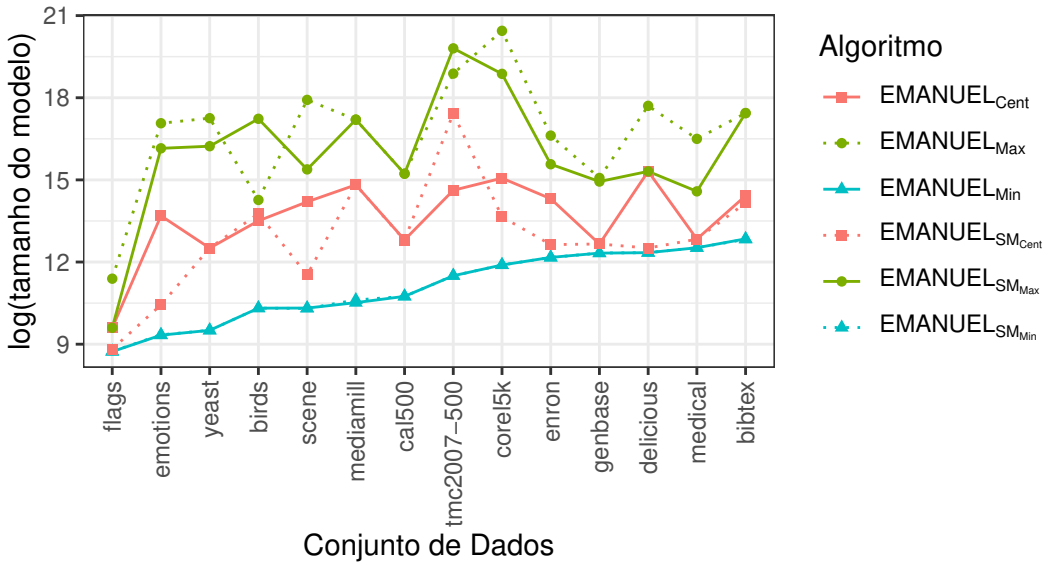


Figura 33 – Tamanho dos modelos induzidos pelos algoritmos selecionados das fronteiras de EMANUEL e EMANUEL$_{SM}$. Os tamanhos dos modelos foram plotados em logaritmo para facilitar a visualização.



Apesar da influência de fatores aleatórios nos resultados de ambas as estratégias, a Figura 32 revela uma tendência de similaridade entre os resultados de EMANUEL e EMANUEL$_{SM}$ quanto ao Macro F-score. Na Figura 32, as linhas que representam os resultados dos algoritmos EMANUEL$_{\text{Max}}$ e EMANUEL$_{\text{SM}_{\text{Max}}}$ seguem uma tendência similar, se sobrepondo em alguns pontos. O mesmo ocorre com os algoritmos EMANUEL$_{\text{Cent}}$ e EMANUEL$_{\text{SM}_{\text{Cent}}}$ e com os algoritmos EMANUEL$_{\text{Min}}$ e EMANUEL$_{\text{SM}_{\text{Min}}}$. Essa tendência reflete uma consistência nos desempenhos obtidos por ambas as estratégias.

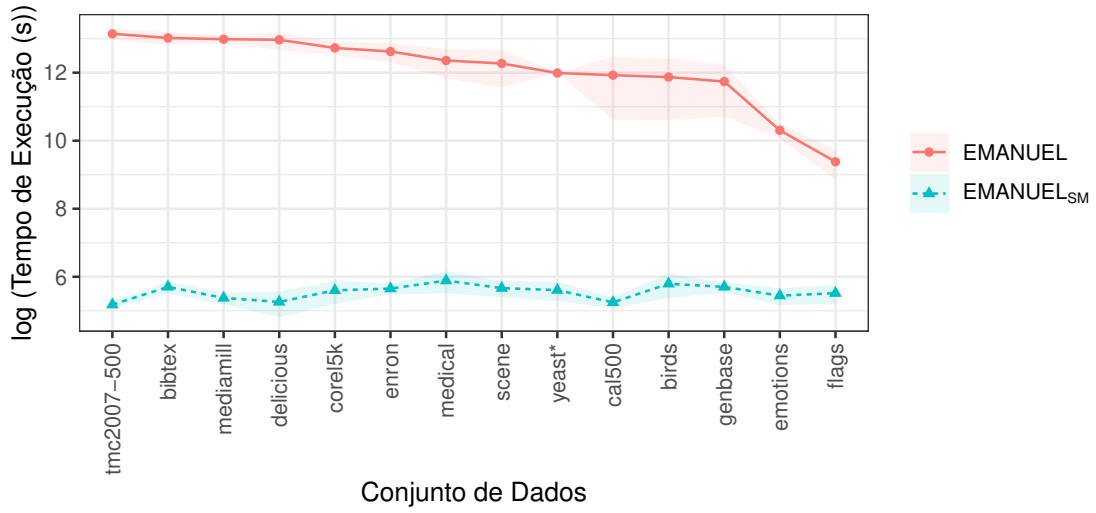
A Figura 33 apresenta os resultados das estratégias para o tamanho do modelo. A mesma tendência observada entre os algoritmos da Figura 32 pode ser observada na Figura 33. Os resultados de EMANUEL$_{\text{Min}}$ e EMANUEL$_{\text{SM}_{\text{Min}}}$ se sobrepõem na Figura 33, indicando que ambas as estratégias produziram modelos com tamanhos semelhantes para esses algoritmos. A mesma tendência é observada entre EMANUEL$_{\text{Cent}}$ e EMANUEL$_{\text{SM}_{\text{Cent}}}$, assim como entre EMANUEL$_{\text{Max}}$ e EMANUEL$_{\text{SM}_{\text{Max}}}$, embora os resultados desses algoritmos apresentem maior variabilidade.

Para confirmar os resultados da Figura 32 e 33, aplicamos o teste não paramétrico de Wilcoxon com nível de significância de 95% ($\alpha = 0.05$) para verificar o desempenho de EMANUEL e EMANUEL$_{SM}$. Para isso, utilizamos os desempenhos dos algoritmos selecionados das fronteiras resultantes de ambas as estratégias. No primeiro teste, a hipótese nula afirmava que as estratégias apresentavam Macro F-scores similares. No segundo teste, a hipótese nula afirmava que os tamanhos dos modelos obtidos pelas estratégias eram semelhantes. No primeiro teste, a hipótese nula foi aceita ($p\_value \approx 0.237$), ou seja, EMANUEL e EMANUEL$_{SM}$ encontraram algoritmos que apresentam Macro

F-scores similares do ponto de vista estatístico. No segundo teste, a hipótese nula foi rejeitada ($p\_value \approx 0.0185$), indicando que os algoritmos encontrados por EMANUEL e EMANUEL$_{SM}$ induzem modelos com tamanhos diferentes. Na maioria dos casos, os modelos induzidos por EMANUEL são maiores que os induzidos por EMANUEL$_{SM}$

As estratégias EMANUEL e EMANUEL$_{SM}$ também foram comparadas em relação ao tempo de execução. A Figura 34 ilustra o gráfico do tempo de execução médio de ambas as estratégias para os conjuntos de dados investigados. Utilizamos o logaritmo do tempo de execução para melhorar a visualização dos resultados, uma vez que os tempos resultantes de ambas as estratégias são discrepantes. Na Figura 34, omitimos o desvio padrão do tempo de execução de EMANUEL para o conjunto de dados *yeast*, pois o tempo de execução de EMANUEL oscilou muito nesse conjunto de dados, sendo maior que a média. Nesse caso, não conseguimos calcular o valor do logaritmo para o desvio padrão abaixo da média (valor negativo). A Figura 81 do Apêndice D apresenta a mesma figura sem escala logarítmica.

Figura 34 – Logaritmo do tempo de execução médio (segundos) de EMANUEL e EMANUEL$_{SM}$ nos conjuntos de dados investigados. O desvio padrão do tempo de execução de EMANUEL no conjunto de dados *yeast* foi omitido no gráfico devido a sua alta oscilação.



O tempo de execução de EMANUEL é superior ao tempo de execução de EMANUEL$_{SM}$. Na estratégia EMANUEL a avaliação dos indivíduos envolve a indução e avaliação dos modelos de classificação. Como consequência, questões como a dimensionalidade dos conjuntos de dados, os algoritmos de classificação selecionados durante a evolução, o número de consultas aos algoritmos de classificação já avaliados e o número de gerações para a convergência do NSGA-II impactam diretamente no tempo de execução da estratégia.

Na estratégia EMANUEL$_{SM}$, não há indução de modelos para a avaliação dos indivíduos. Em vez disso, os modelos substitutos preveem os valores dos objetivos, tornando

a execução muito mais rápida. O tempo de execução de EMANUEL$_{SM}$ é determinado pelo tempo das previsões e pelo número de gerações até a convergência do NSGA-II. A Figura 34 mostra a diferença no tempo de execução entre as duas estratégias para cada conjunto de dados. O tempo de execução do EMANUEL$_{SM}$ foi significativamente inferior ao tempo de execução de EMANUEL em todos os conjuntos de dados, permanecendo abaixo de 460 segundos em todos os conjuntos de dados.

Dessa forma, confirmamos a nossa hipótese de pesquisa. As estratégias EMANUEL e EMANUEL$_{SM}$ apresentam hipervolumes semelhantes, os Macro F-scores dos algoritmos selecionados das fronteiras de ambas as estratégias são estatisticamente similares, os tamanhos dos modelos são estatisticamente diferentes, mas na maioria dos casos EMANUEL$_{SM}$ induz modelos menores, e o tempo de execução de EMANUEL$_{SM}$ é significativamente inferior ao de EMANUEL.

6.6.3 Comparação das Estratégias Propostas com os *Baselines*

Comparamos os três algoritmos selecionados das fronteiras de Pareto de EMANUEL e de EMANUEL$_{SM}$ com os oito algoritmos *baselines* descritos na Seção 6.5. As comparações foram feitas em relação ao Macro F-score e ao tamanho dos modelos induzidos, nas Seções 6.6.3.1 e 6.6.3.2, respectivamente. Para confirmar os resultados discutidos em relação aos objetivos, a Seção 6.6.3.3 traz os testes estatísticos.

6.6.3.1 Macro F-score

Os algoritmos do experimento foram comparados em relação ao Macro F-score. A Figura 35 apresenta os resultados dos algoritmos para essa medida. Na Figura 35, os algoritmos com maiores Macro F-scores na maioria dos conjuntos de dados são EMANUEL$_{\text{Max}}$ e EMANUEL$_{\text{SM}_{\text{Max}}}$. Os algoritmos com menores Macro F-scores, majoritariamente, são EMANUEL$_{\text{Min}}$ e EMANUEL$_{\text{SM}_{\text{Min}}}$. Os demais algoritmos apresentaram Macro F-scores entre os valores mínimos e máximos alcançados pelos algoritmos supracitados.

Adicionalmente, a Figura 36 apresenta o *ranking* dos algoritmos avaliados no experimento em relação ao Macro F-score, para todos os conjuntos de dados. No eixo y, os algoritmos são apresentados em ordem crescente de Macro F-score, de acordo com a classificação média. Quanto mais azuis os quadrados, menor o *ranking*, ou seja, melhores os resultados para o Macro F-score. A Figura 36 confirma os resultados relatados anteriormente, EMANUEL$_{\text{Max}}$ e EMANUEL$_{\text{SM}_{\text{Max}}}$, e EMANUEL$_{\text{Min}}$ e EMANUEL$_{\text{SM}_{\text{Min}}}$, são os algoritmos com os melhores e os piores valores para o Macro F-score, respectivamente. Além disso, na Figura 36, `Auto-MEKA`, EMANUEL$_{\text{Cent}}$ e EMANUEL$_{\text{SM}_{\text{Cent}}}$ também estão entre os algoritmos com melhores *rankings*.

EMANUEL$_{\text{Max}}$, EMANUEL$_{\text{SM}_{\text{Max}}}$, EMANUEL$_{\text{Cent}}$ e EMANUEL$_{\text{SM}_{\text{Cent}}}$ são os algoritmos selecionados das fronteiras de Pareto de EMANUEL e de EMANUEL$_{SM}$ com melho-

Figura 35 – Macro F-score dos algoritmos selecionados das fronteiras de EMANUEL e de EMANUEL$_{SM}$, e Macro F-score médio dos algoritmos *baselines*.

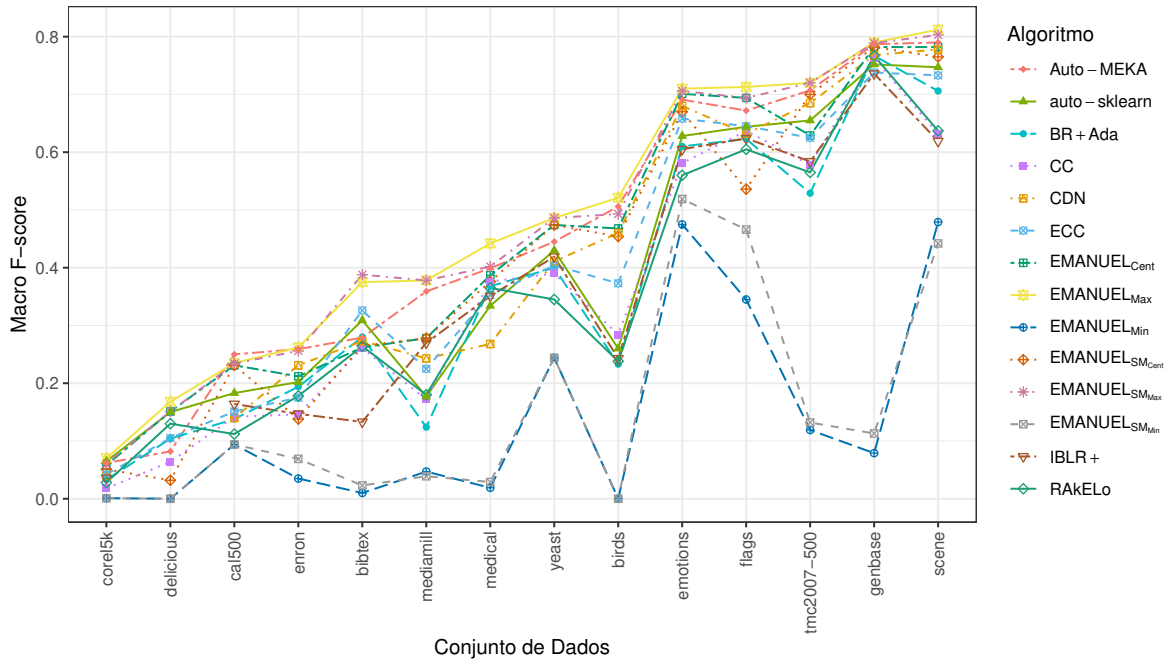


Figura 36 – *Ranking* dos algoritmos avaliados no experimento quanto ao Macro F-score. Na figura, *rankings* iguais para os algoritmos de um conjunto de dados representam algoritmos que tiveram Macro F-scores iguais.

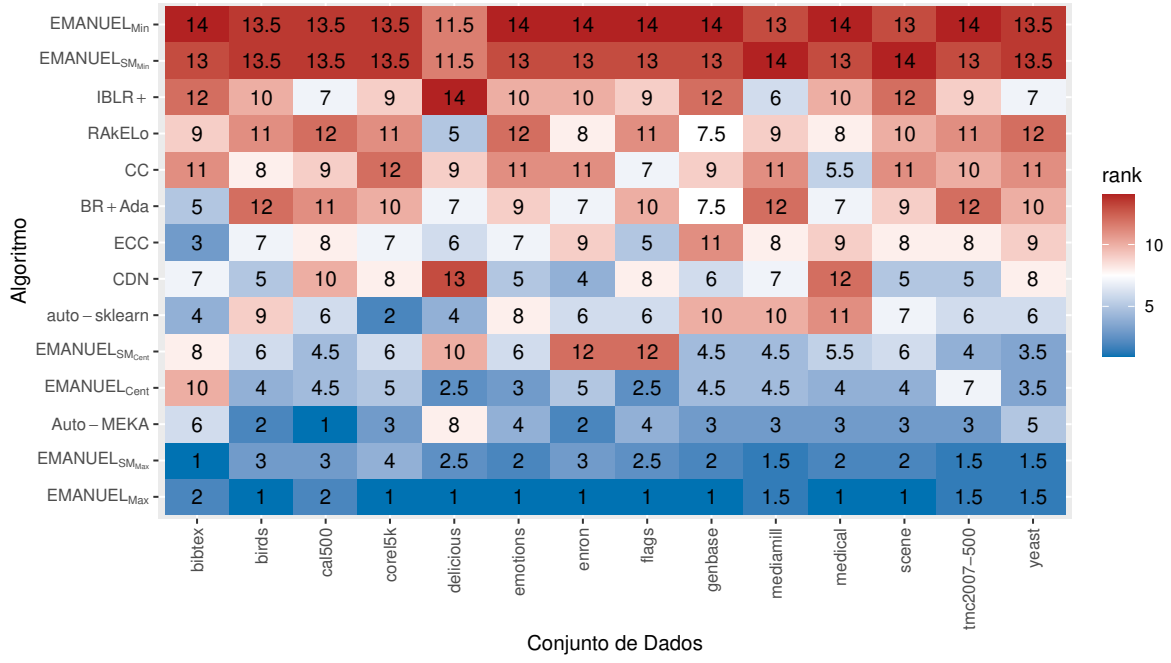
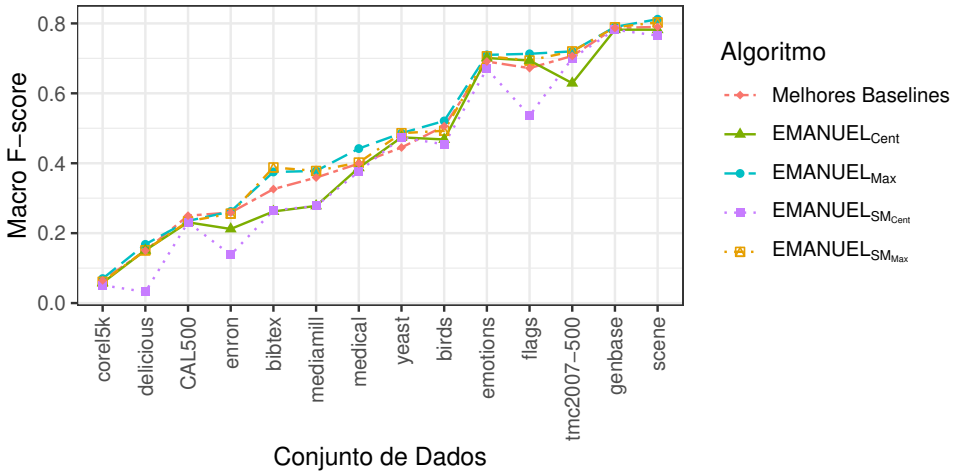


Figura 37 – Macro F-score dos modelos induzidos por $\text{EMANUEL}_{\text{Max}}$, $\text{EMANUEL}_{\text{SM}_{\text{Max}}}$, $\text{EMANUEL}_{\text{Cent}}$, $\text{EMANUEL}_{\text{SM}_{\text{Cent}}}$ e pelos melhores algoritmos *baselines* de cada conjunto de dados.



res *rankings* (Figura 36). Outra possível análise é a comparação desses algoritmos com os melhores algoritmos *baselines* de cada conjunto de dados. Na Figura 37, $\text{EMANUEL}_{\text{Max}}$ e $\text{EMANUEL}_{\text{SM}_{\text{Max}}}$ apresentaram Macro F-scores superiores aos melhores algoritmos *baselines* na maioria dos conjuntos de dados. No entanto, os melhores algoritmos *baselines* tiveram Macro F-scores superiores a $\text{EMANUEL}_{\text{Cent}}$ e $\text{EMANUEL}_{\text{SM}_{\text{Cent}}}$ na maioria das vezes.

6.6.3.2 Tamanho dos Modelos

Nesta seção comparamos os algoritmos do experimento em relação ao tamanho dos modelos. Esses resultados são apresentados na Figura 38. Nessa Figura, não há um padrão definido para os algoritmos com maiores tamanhos de modelos. Algoritmos como `Auto-MEKA`, CDN e ECC tendem a induzir modelos maiores. Já os algoritmos com os menores tamanhos de modelos são majoritariamente $\text{EMANUEL}_{\text{Min}}$ e $\text{EMANUEL}_{\text{SM}_{\text{Min}}}$. Na maioria das vezes, os demais algoritmos apresentaram tamanhos de modelos superiores aos valores mínimos alcançados. Em complemento à Figura 38, a Figura 39 mostra o *ranking* dos algoritmos avaliados no experimento em relação ao tamanho dos modelos, para todos os conjuntos de dados. De acordo com a Figura 39, CDN, ECC e `Auto-MEKA` são os algoritmos que induzem os maiores modelos, ao passo que $\text{EMANUEL}_{\text{Min}}$ e $\text{EMANUEL}_{\text{SM}_{\text{Min}}}$ induzem os menores. Assim, a Figura 39 confirma as análises prévias em relação ao tamanho dos modelos.

De acordo com a Figura 39, os algoritmos selecionados das fronteiras de Pareto de EMANUEL e de EMANUEL_{SM} com os melhores *rankings* para o tamanho do modelo foram $\text{EMANUEL}_{\text{Min}}$, $\text{EMANUEL}_{\text{SM}_{\text{Min}}}$, $\text{EMANUEL}_{\text{SM}_{\text{Cent}}}$ e $\text{EMANUEL}_{\text{Cent}}$. Para estender

Figura 38 – Tamanho dos modelos dos algoritmos selecionados das fronteiras de EMANUEL e de EMANUEL$_{SM}$, e tamanhos dos modelos médios dos algoritmos *baselines*.

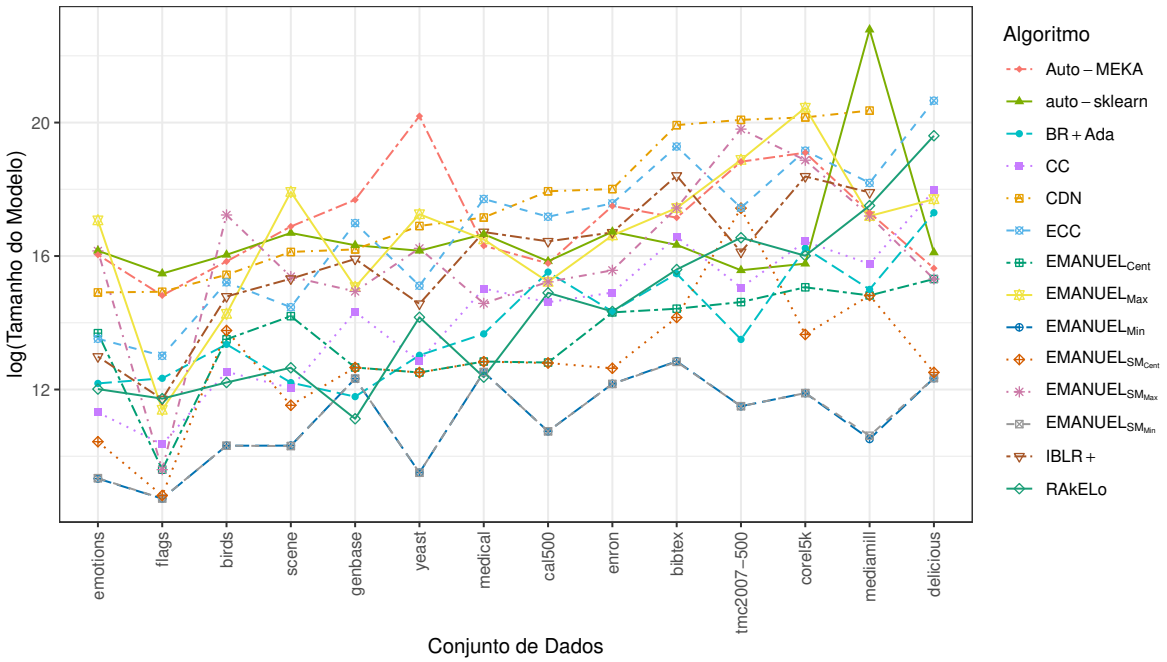


Figura 39 – *Ranking* dos algoritmos avaliados no experimento quanto ao tamanho do modelo. Na figura, *rankings* iguais para os algoritmos de um conjunto de dados representam algoritmos que tiveram tamanhos de modelos iguais.

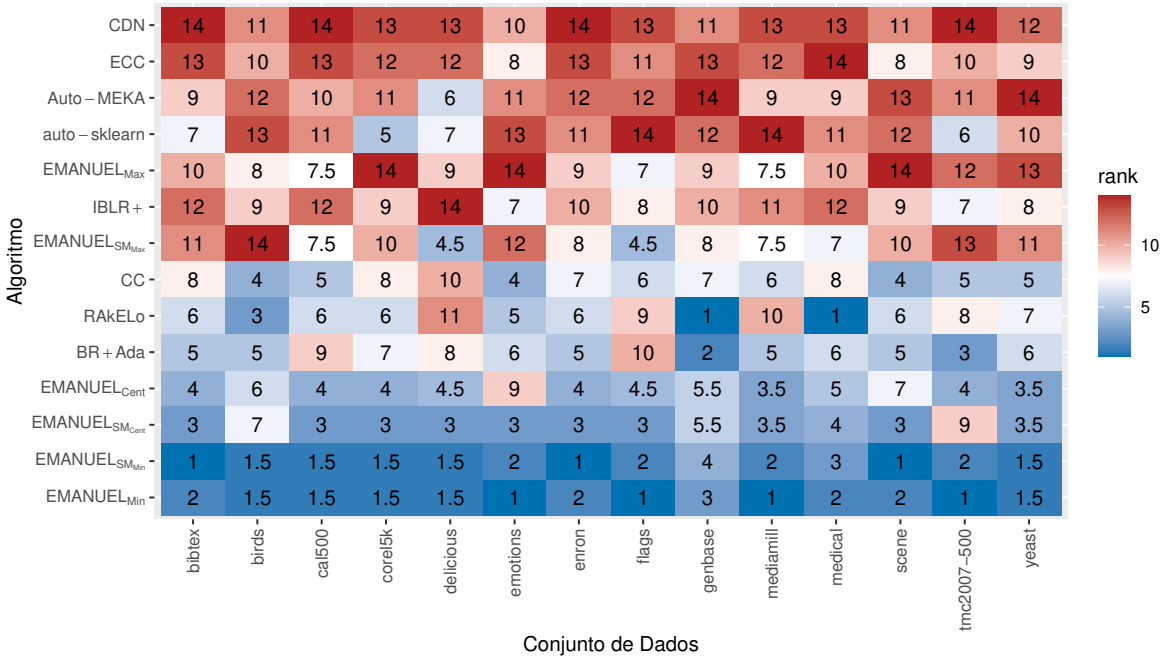
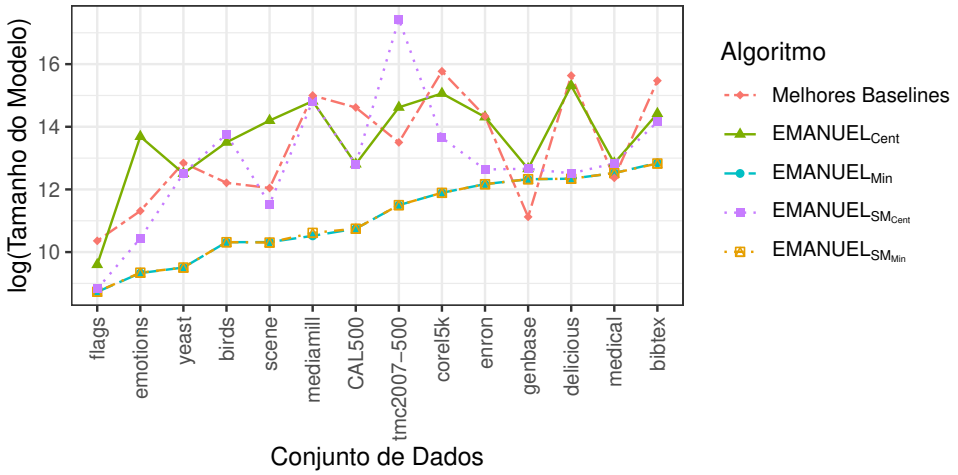


Figura 40 – Tamanhos dos modelos induzidos por $\text{EMANUEL}_{\text{Min}}$, $\text{EMANUEL}_{\text{SM}_{\text{Min}}}$, $\text{EMANUEL}_{\text{Cent}}$, $\text{EMANUEL}_{\text{SM}_{\text{Cent}}}$ e pelos melhores algoritmos *baselines* de cada conjunto de dados.



as análises, comparamos esses algoritmos com os melhores algoritmos *baselines* de cada conjunto de dados. Na Figura 40, $\text{EMANUEL}_{\text{Min}}$ e $\text{EMANUEL}_{\text{SM}_{\text{Min}}}$ induziram os menores modelos em 12 dos 14 conjuntos de dados. Os melhores *baselines* de cada conjunto de dados induziram modelos com tamanhos superiores à $\text{EMANUEL}_{\text{Min}}$ e $\text{EMANUEL}_{\text{SM}_{\text{Min}}}$, a exceção ocorreu nos conjuntos de dados *genbase* e *medical*. Os melhores *baselines* também induziram modelos com tamanhos superiores à $\text{EMANUEL}_{\text{Cent}}$ e $\text{EMANUEL}_{\text{SM}_{\text{Cent}}}$ na maioria das vezes.

6.6.3.3 Teste Estatístico

Para confirmar os resultados discutidos nas Seções 6.6.3.1 e 6.6.3.2, realizamos testes estatísticos para identificar os grupos de algoritmos com os melhores desempenhos. Os algoritmos CDN e IBLR+ não concluíram suas execuções em tempo adequado (sete dias) para o conjunto de dados *delicious* e, portanto, não produziram resultados. Assim, removemos esse conjunto de dados dos testes estatísticos para evitar viés nas análises.

O teste de Friedman foi aplicado para verificar se os algoritmos avaliados apresentavam desempenhos semelhantes em relação aos objetivos. A hipótese nula foi rejeitada no teste para o Macro F-score. O teste *post-hoc* de Nemenyi foi então utilizado para identificar os grupos de algoritmos com Macro F-scores estatisticamente diferentes. No diagrama de diferença crítica da Figura 41, os algoritmos com melhores classificações no *ranking* apresentam os melhores Macro F-scores, enquanto os algoritmos com classificações piores obtiveram desempenhos piores.

A classificação dos algoritmos com melhores Macro F-scores foi: $\text{EMANUEL}_{\text{Max}}$, $\text{EMANUEL}_{\text{SM}_{\text{Max}}}$, `Auto-MEKA`, $\text{EMANUEL}_{\text{Cent}}$ e $\text{EMANUEL}_{\text{SM}_{\text{Cent}}}$. Do ponto de vista

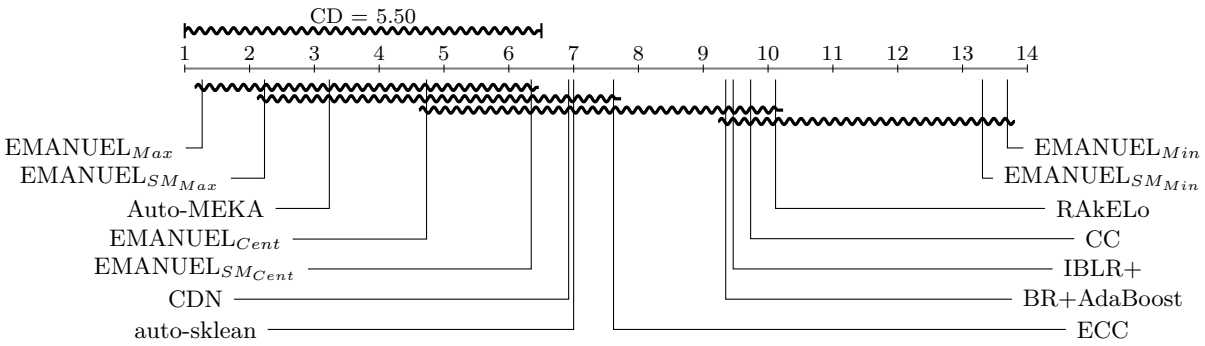


Figura 41 – Diagrama de diferença crítica dos algoritmos avaliados em relação ao Macro F-score.

estatístico, os algoritmos supracitados apresentaram desempenhos semelhantes. Os algoritmos desse conjunto foram estatisticamente superiores aos demais algoritmos avaliados (CDN, `auto-sklearn`, ECC, BR com AdaBoost, IBLR+, CC, RAkELo, EMANUEL$_{\text{SM}_{\text{Min}}}$, EMANUEL$_{\text{Min}}$). Além disso, ainda na Figura 41, algoritmos como EMANUEL$_{\text{SM}_{\text{Max}}}$, EMANUEL$_{\text{Cent}}$ e EMANUEL$_{\text{SM}_{\text{Cent}}}$ possuem desempenhos equivalentes aos desempenhos dos algoritmos CDN, `auto-sklearn` e ECC. Os resultados indicam que as fronteiras de Pareto resultantes de EMANUEL e de EMANUEL$_{SM}$ contêm algoritmos com desempenhos equivalentes a outras estratégias AutoML e a algoritmos que comumente possuem bons resultados para a classificação multirrótulo (BOGATINOVSKI et al., 2022; GARCÍA-PEDRAJAS et al., 2024).

No teste estatístico para o tamanho do modelo, a hipótese nula também foi rejeitada, ou seja, os tamanhos dos modelos treinados foram diferentes. A partir do teste *post-hoc* de Nemenyi, identificamos os grupos de algoritmos com tamanhos de modelos diferentes. No diagrama de diferença crítica da Figura 42, os algoritmos com melhores posições no *ranking*, ou com os menores tamanhos de modelos, foram os algoritmos EMANUEL$_{\text{Min}}$, EMANUEL$_{\text{SM}_{\text{Min}}}$, EMANUEL$_{\text{SM}_{\text{Cent}}}$, EMANUEL$_{\text{Cent}}$, BR com Adaboost, RAkELo e CC. Os demais algoritmos apresentaram tamanhos de modelos estatisticamente superiores (EMANUEL$_{\text{SM}_{\text{Max}}}$, IBLR+, EMANUEL$_{\text{Max}}$, auto-sklearn, ECC, `Auto-MEKA`, e CDN). Mais uma vez, os resultados apontam que as fronteiras de Pareto resultantes de EMANUEL e de EMANUEL$_{SM}$ possuem algoritmos que induzem modelos de diferentes tamanhos. No entanto, modelos com tamanhos menores normalmente possuem Macro F-scores inferiores.

EMANUEL$_{\text{Cent}}$ e EMANUEL$_{\text{SM}_{\text{Cent}}}$ são os algoritmos de classificação multirrótulo com os melhores resultados para ambos os objetivos, pois eles pertencem aos grupos de soluções com melhores desempenhos nas Figuras 41 e 42. Esses algoritmos são os que melhor ponderam ambos os objetivos da otimização. Os algoritmos EMANUEL$_{\text{Max}}$, EMANUEL$_{\text{SM}_{\text{Max}}}$, `Auto-MEKA`, CDN e `auto-sklearn` estão no grupo de algoritmos com os melhores Macro F-scores na Figura 41. Porém, na Figura 42, esses algoritmos se encontram entre os que apresentam os maiores tamanhos de modelos. Os algoritmos EMANUEL$_{\text{Min}}$, EMANUEL$_{\text{SM}_{\text{Min}}}$

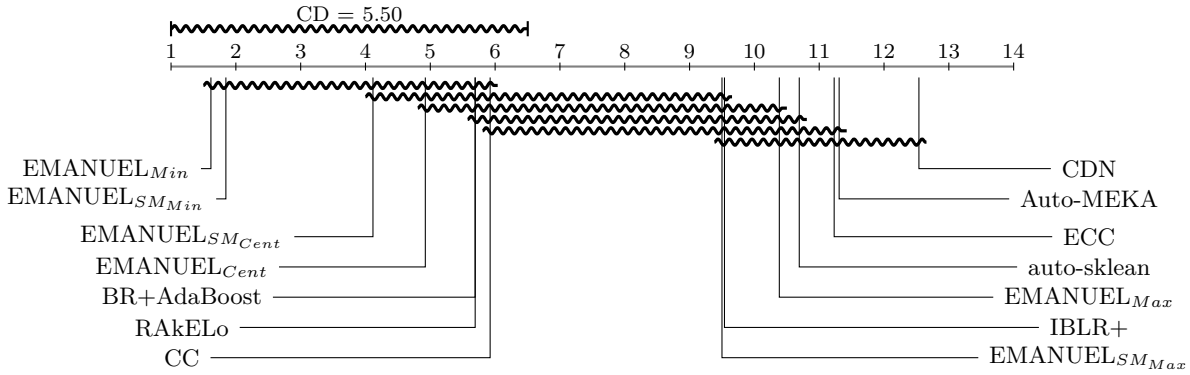


Figura 42 – Diagrama de diferença crítica dos algoritmos avaliados em relação ao tamanho do modelo.

apresentam os melhores tamanhos de modelos e os piores Macro F-scores nas Figuras 42 e 41, respectivamente. Assim, os testes estatísticos confirmam os resultados apresentados nas Seções 6.6.3.1 e 6.6.3.2. O Apêndice D apresenta os resultados de Macro F-score e do tamanho do modelo obtidos pelos algoritmos avaliados neste capítulo.

6.7 Considerações Finais

Neste capítulo apresentamos EMANUEL$_{SM}$, uma estratégia AutoML baseada em EMANUEL. A nova estratégia foi desenvolvida nos moldes de EMANUEL, mas ela não induz modelos de classificação para o cálculo dos valores dos objetivos. Ela prevê o desempenho (Macro F-score) e o tamanho do modelo induzido por um algoritmo de classificação multirrótulo utilizando modelos substitutos.

Comparamos os resultados de EMANUEL e de EMANUEL$_{SM}$. A fronteira resultante de EMANUEL$_{SM}$ possui menos algoritmos de classificação multirrótulo do que a fronteira de EMANUEL. No entanto, os hipervolumes resultantes das fronteiras das estratégias são semelhantes. Além disso, os Macro F-scores dos algoritmos selecionados das fronteiras de ambas as estratégias são estatisticamente similares e os tamanhos dos modelos induzidos por ambas as estratégias são estatisticamente diferentes. Logo, EMANUEL$_{SM}$ pode encontrar algoritmos com desempenho equivalente ao desempenho de EMANUEL, mas que induzem modelos menores. A maior diferença entre EMANUEL e EMANUEL$_{SM}$ está no tempo de execução, que é significativamente menor em EMANUEL$_{SM}$. Esses resultados confirmam nossa hipótese de pesquisa.

Também comparamos EMANUEL e EMANUEL$_{SM}$ com os algoritmos *baselines*. Em relação ao Macro F-score, EMANUEL$_{\text{Max}}$, EMANUEL$_{\text{SM}_{\text{Max}}}$, `Auto-MEKA`, EMANUEL$_{\text{Cent}}$ e EMANUEL$_{\text{SM}_{\text{Cent}}}$ foram estatisticamente superiores aos demais algoritmos avaliados. Os tamanhos dos modelos de EMANUEL$_{\text{Min}}$, EMANUEL$_{\text{SM}_{\text{Min}}}$, EMANUEL$_{\text{SM}_{\text{Cent}}}$, EMANUEL$_{\text{Cent}}$, BR com Adaboost, RAkELo e CC foram estatisticamente menores que os demais algoritmos. EMANUEL$_{\text{Cent}}$ e EMANUEL$_{\text{SM}_{\text{Cent}}}$ pertenceram aos grupos de

melhores algoritmos para ambos os objetivos, ou seja, são algoritmos com bons resultados para Macro F-score e para o tamanho do modelo.

Os resultados demonstram que os algoritmos selecionados das fronteiras de EMANUEL e de EMANUEL$_{SM}$ apresentam desempenhos estatisticamente semelhantes. No entanto, a vantagem de EMANUEL$_{SM}$ é o seu tempo de execução inferior. A fim de explorar ainda mais o AutoML para classificação multirrótulo e de buscar de resultados ainda melhores, podemos considerar outros requisitos para o AutoML. O próximo capítulo estuda a seleção automática de atributos no contexto do AutoML. Trata-se de um estudo inicial, com hiperespaço reduzido e otimização mono-objetivo, base para a extensão de EMANUEL.

Capítulo 7

Seleção Automática de Atributos em Conjunto de Dados Multirrótulo

Nos Capítulos 5 e 6, exploramos AutoML para classificação multirrótulo, automatizando a busca por modelos de classificação. As demais tarefas *pipeline* não foram abordadas até então. Neste capítulo, estamos interessados na automatização da tarefa *pipeline* engenharia de atributos, especificamente na seleção de atributos. A seleção de atributos é um campo da engenharia de atributos que visa encontrar dentre os atributos originais um subconjunto de atributos relevantes e sem redundância (HE; ZHAO; CHU, 2021; ZÖLLER; HUBER, 2021). Os atributos selecionados devem reter as características originais dos dados, mas com uma dimensionalidade menor, podendo acelerar o aprendizado dos algoritmos do AM, conservando ou mesmo melhorando o desempenho desses algoritmos.

Este capítulo investiga o uso de uma estratégia de seleção de atributos automatizada para problemas de classificação multirrótulo, denominada *Automated Multi-label Feature Selection* (AutoMLFS). AutoMLFS utiliza o algoritmo ECC como algoritmo de classificação multirrótulo base. O algoritmo de otimização avalia diferentes algoritmos de seleção de atributos multirrótulo e diferentes números de atributos a serem selecionados. Um modelo ECC é induzido com os atributos selecionados. O objetivo da estratégia é encontrar o algoritmo de seleção de atributos multirrótulo e o número de atributos que possibilitam a indução de modelos de AM mais precisos. Nossa hipótese de pesquisa é que a nova estratégia é capaz de encontrar modelos que tenham resultados superiores, utilizando menos atributos, aos modelos induzidos com atributos selecionados por algoritmos específicos.

Este capítulo está organizado da seguinte forma: a Seção 7.1 faz uma breve revisão sobre a seleção de atributos em conjuntos de dados multirrótulo; a Seção 7.2 explica o funcionamento da estratégia AutoMLFS; as Seções 7.3, 7.4 e 7.5 apresentam o hiperes-

paço, questões metodológicas relacionadas à implementação e ao *setup* de execução, e aos *baselines* de comparação, respectivamente; a Seção 7.6 discute os resultados; e a Seção 7.7 conclui o capítulo com as considerações finais.

7.1 Seleção de Atributos Multirrótulo

Dados os conjuntos de dados de entrada ($\mathbf{X}$) e de saída ($\mathbf{Y}$), um algoritmo de seleção de atributos deve encontrar no conjunto de entrada um subconjunto de atributos não redundantes e com relevância para o conjunto de saída (KASHEF; NEZAMABADI-POUR; NIKPOUR, 2018). Segundo Kashef, Nezamabadi-pour e Nikpour (2018), a seleção de atributos multirrótulo pode ser categorizada sob diferentes perspectivas. Na perspectiva da interação com o algoritmo de aprendizado, há três abordagens para a seleção de atributos: filtro; *wrapper*; e *embedded*.

A abordagem de filtro é independente do algoritmo de AM, ela utiliza as características dos atributos para ranqueá-los com base em um critério pré-estabelecido e seleciona os atributos com classificações mais altas. Algoritmos de seleção de atributos baseados em medidas de importância, como a Estatística Qui-quadrado, Ganho de Informação, Informação Mútua e ReliefF, são exemplos de algoritmos de filtro para problemas monorrótulo. No entanto, esses algoritmos podem ser adaptados para problemas multirrótulo.

A abordagem *wrapper* emprega mecanismos de busca que selecionam e avaliam iterativamente um subconjunto de atributos até que um critério de parada seja satisfeito. Essa abordagem utiliza seleção sequencial de atributos (*sequential backward elimination* e *sequential forward selection*, por exemplo) e heurísticas de busca como algoritmos genéticos, por exemplo. Na abordagem *embedded*, a seleção de atributos e o treinamento do algoritmo de aprendizado ocorrem em um processo único, onde os melhores atributos são utilizados na indução do modelo. Classificadores baseados em árvores são exemplos da abordagem *embedded*.

Na perspectiva do formato dos dados, a seleção de atributos multirrótulo é categorizada nas abordagens de transformação de problemas e de adaptação de algoritmos (KASHEF; NEZAMABADI-POUR; NIKPOUR, 2018), essas abordagens são as mesmas abordagens da classificação multirrótulo (TSOUMAKAS; KATAKIS; VLAHAVAS, 2009). As Seções 7.1.1 e 7.1.2 descrevem ambas as abordagens.

7.1.1 Transformação de Problemas

Na abordagem transformação de problemas, dados multirrótulo são decompostos em monorrótulo e algoritmos de seleção monorrótulo são utilizados. A transformação *Binary Relevance* (BR) converte o conjunto de rótulos ($\mathbf{Y}$) em q conjuntos de rótulos binários, onde q é o número de rótulos do problema original (TSOUMAKAS; KATAKIS;

VLAHAVAS, 2009; KASHEF; NEZAMABADI-POUR; NIKPOUR, 2018). Em seguida, um algoritmo de seleção de atributos monorrótulo é utilizado para ranquear os atributos para cada um dos q conjuntos de rótulos binários. Finalmente, os múltiplos *ranks* são agregados para obter um subconjunto de atributos. A transformação BR é simples, porém não considera a correlação entre os rótulos.

Outra alternativa é a transformação *Label Powerset* (LP). Essa transformação cria um novo conjunto de rótulos (**Y'**) contendo todas as possíveis combinações de rótulos do conjunto de rótulos original (**Y**), criando um problema de classificação multiclasse (TSOUMAKAS; KATAKIS; VLAHAVAS, 2009; KASHEF; NEZAMABADI-POUR; NIKPOUR, 2018). Após a transformação LP, a seleção de atributos ocorre com um algoritmo monorrótulo, utilizando o novo conjunto de rótulos (**Y'**). O resultado da seleção de atributos monorrótulo e multirrótulo é o mesmo. Essa transformação considera a correlação dos rótulos, no entanto, o número de rótulos gerados nessa abordagem é alto e pode haver rótulos sem instâncias. A abordagem *Pruned Problem Transformation* (PPT) é uma extensão da abordagem LP. PPT poda do novo conjunto de rótulos (**Y'**) aqueles com frequência inferior a um limiar (*threshold*), eliminando ou atribuindo instâncias a outra classe.

No geral, alguns estudos da literatura exploram essas transformações. Wieczorkowska, Synak e Raś (2006) apresentaram um algoritmo de seleção de atributos multirrótulo com a transformação LP e a Estatística Qui-quadrado para ranquear os atributos. Doquire e Verleysen (2011) fizeram a seleção de atributos multirrótulo transformando problemas multirrótulo com PPT e calculando a importância dos atributos com Informação Mútua. Spolaôr et al. (2013) propuseram quatro algoritmos de seleção de atributos multirrótulo, utilizando as transformações BR e LP combinadas com as medidas ReliefF e Ganho de Informação. Ao explorar algoritmos baseados na transformação BR, alguns autores estimam a importância do atributo (*rank*) calculando a média para cada rótulo (SPOLAÔR et al., 2013).

7.1.2 Adaptação de Algoritmos

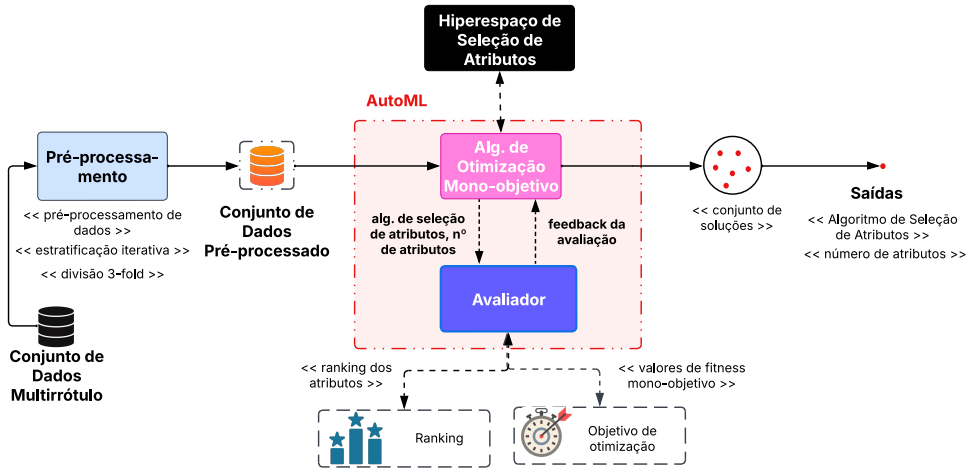
A abordagem adaptação de algoritmos aprimora os algoritmos de seleção de atributos monorrótulo para lidar com um número maior de rótulos. Li et al. (2014) propuseram um algoritmo baseado no Ganho de Informação denominado *Information Gain for Multi-label Feather* (IGMF). IGMF calcula o Ganho de Informação entre cada atributo e o conjunto de rótulos e, em seguida, adota um limiar para selecionar o subconjunto de atributos ótimos. Lee e Kim (2017) apresentaram o algoritmo *Scalable Criterion for Large Label Set* (SCLS). SCLS calcula uma pontuação de avaliação global a partir dos atributos e de todos os rótulos, o que permite identificar os principais atributos. Essa pontuação é escalável e funciona bem ao lidar com conjuntos de dados com muitos rótulos.

Zhang, Liu e Gao (2019) apresentaram o algoritmo *Label Redundancy Feature Selection* (LRFS), que avalia a importância de um atributo utilizando informação condicional mútua e redundância de rótulos. Zhang et al. (2021) propuseram dois algoritmos de seleção de atributos multirrótulo: *Multi-label Feature Selection considering Label Supplementation* (LSMFS) e *Multi-label Feature Selection considering Maximum Label Supplementation* (MLSMFS). LSMFS usa uma medida chamada *Additional Information* que captura as informações dos atributos sob a condição de que os rótulos têm relações de suplementação. MLSMFS utiliza a medida *Maximum Additional Information* que identifica o rótulo que mais contribui para o atributo referente à suplementação do rótulo.

7.2 Como o AutoMLFS trabalha?

O funcionamento de AutoMLFS é ilustrado na Figura 43. Inicialmente, o conjunto de dados multirrótulo passa pela etapa de pré-processamento manual. Nesta etapa, rótulos associados a poucas instâncias, atributos com os mesmos valores para todas as instâncias, instâncias não rotuladas e duplicadas são removidos do conjunto de dados. Os dados são subdivididos por meio de validação cruzada estratificada *3-folds*. Esse conjunto de dados resultante do pré-processamento manual é a entrada para a estratégia AutoML.

Figura 43 – Fluxograma do funcionamento da estratégia AutoMLFS.



AutoMLFS utiliza um algoritmo de otimização mono-objetivo para seleção automatizada dos melhores atributos. O algoritmo de otimização obtém do hiperespaço o algoritmo de seleção de atributos multirrótulo e o número de atributos a serem selecionados. No entanto, esse algoritmo não é executado no AutoML. Em vez disso, AutoMLFS mantém os *rankings* dos atributos do conjunto de dados em questão, para todos os algoritmos de seleção de atributos do hiperespaço. Os *rankings* são pré-calculados para evitar a execução dos algoritmos de seleção de atributos repetidas vezes no AutoML.

O algoritmo de otimização envia o algoritmo de seleção de atributos multirrótulo e o número de atributos a serem selecionados do conjunto de dados para o avaliador. O avaliador consulta o *ranking* do algoritmo em avaliação e filtra os atributos com os melhores *ranks*, limitado ao número de atributos a serem selecionados. Assim, um novo conjunto de dados é criado, contendo apenas os atributos de interesse. O avaliador treina um classificador multirrótulo nesse novo conjunto de dados.

A partir das previsões do classificador, o avaliador calcula o valor do objetivo e retorna esse valor ao algoritmo de otimização. O processo de otimização ocorre por um determinado número de iterações e, em cada iteração, calcula-se o *fitness* do modelo induzido em um conjunto de dados reduzido. O resultado do algoritmo de otimização é o algoritmo de seleção de atributos multirrótulo e o número de atributos a serem selecionados, os quais permitem a criação de um conjunto de dados com um número reduzido de atributos para a indução de um classificador que otimiza a função objetivo.

Durante a execução, o algoritmo de otimização pode encontrar múltiplas soluções que maximizam/minimizam o valor do objetivo. Por isso, na Figura 7.2, o resultado de AutoMLFS é um conjunto de soluções em vez de uma única solução. Desse conjunto de soluções, escolhemos a solução cujo algoritmo de seleção de atributos utiliza o menor número de atributos, mantendo o desempenho preditivo.

Neste experimento, o algoritmo de otimização foi a OB com Processos Gaussianos. A OB constrói um modelo probabilístico para uma função f que atua como um modelo substituto (BROCHU; CORA; FREITAS, 2010; FEURER; SPRINGENBERG; HUTTER, 2015). Em cada iteração, uma função de aquisição usa as estimativas do modelo para explorar o espaço de interesse e determinar o ponto mais promissor. Uma vez definido o melhor ponto, o modelo probabilístico é atualizado. A OB é um algoritmo de otimização eficiente, pois a função de aquisição direciona a busca para áreas do espaço de interesse com maior probabilidade de produzir os melhores resultados (BROCHU; CORA; FREITAS, 2010). Adicionalmente, o uso de modelos gaussianos como modelo substituto fornece boas previsões para espaços de entrada numérica e de baixa dimensão (FEURER; SPRINGENBERG; HUTTER, 2015). O objetivo da otimização foi maximizar o Macro F-score. No entanto, para trabalhar com um problema de minimização na OB, adotamos minimizar $-1 * \text{Macro F-score}$. O algoritmo de classificação multirrótulo empregado foi o ECC, pois ele normalmente induz modelos de alto desempenho para problemas multirrótulo (MOYANO et al., 2018; BOGATINOVSKI et al., 2022).

7.3 Hiperespaço

O hiperespaço de AutoMLFS contém algoritmos de seleção de atributos multirrótulo e os possíveis números de atributos a serem selecionados. Adaptamos os algoritmos de seleção de atributos monorrótulo SelectKBest, *Recursive Feature Elimination* (RFE) e

SelectFromModel da biblioteca `scikit-learn` para lidar com problemas multirrótulo e desenvolvemos algoritmos baseados na função de avaliação de atributos ReliefF.

A Tabela 8 apresenta os algoritmos de seleção de atributos multirrótulo do hiperespaço, bem como seus hiperparâmetros, as classificações desses algoritmos em relação aos algoritmos de aprendizado e em relação à interação com os dados. Os algoritmos da abordagem transformação de problemas utilizam as transformações BR e PPT, descritas na Seção 7.1.1.

Tabela 8 – Algoritmos de seleção de atributos multirrótulo do hiperespaço. Na tabela, TP e AA são as abordagens Transformação de Problemas e Adaptação de Algoritmos, respectivamente.

#	**Algoritmo**	**Hiperparâmetros**	**Interação com Algoritmo de AM**	**Interação com os Dados**
1	BRSelectKBest	method = Estatística qui-quadrado	Filtro	TP (BR)
2	BRSelectKBest	method = Informação mútua	Filtro	TP (BR)
3	BRSelectKBest	method = F-value da ANOVA	Filtro	TP (BR)
4	PPTSelectKBest	method = Estatística qui-quadrado	Filtro	TP (PPT)
5	PPTSelectKBest	method = Informação mútua	Filtro	TP (PPT)
6	PPTSelectKBest	method = F-value da ANOVA	Filtro	TP (PPT)
7	BRReliefF	-	Filtro	TP (BR)
8	PPTReliefF	-	Filtro	TP (PPT)
9	MLC_RFE	classifier = *Random Forest*	*Embedded*	AA
10	PPT_RFE	classifier = *Random Forest*	*Embedded*	TP (PPT)
11	MLCSelectFromModel	classifier = *Random Forest*	*Embedded*	AA
12	PPTSelectFromModel	classifier = *Random Forest*	*Embedded*	TP (PPT)

O hiperespaço é composto por 12 algoritmos de seleção de atributos multirrótulo, uma vez que os algoritmos BRSelectKBest e PPTSelectKBest podem ser utilizados com a Estatística Qui-quadrado, a Informação Mútua e o F-value da ANOVA. Para cada algoritmo do hiperespaço, os hiperparâmetros foram fixados, não havendo otimização de hiperparâmetros no AutoML. Os algoritmos da Tabela 8 são classificados como filtro e *embedded* quanto à interação com os algoritmos de aprendizado e das abordagens transformação de problemas e adaptação de algoritmos quanto à interação com os dados. Para os algoritmos baseados na transformação PPT, adotamos o `limiar` igual a seis. O Apêndice E faz uma breve descrição dos algoritmos do hiperespaço.

Todos os algoritmos do hiperespaço produzem um *score* que representa a importância dos atributos. Assim, é possível encontrar o *ranking* desses atributos. Para cada conjunto de dados e cada algoritmo do hiperespaço, pré-calculamos o *ranking* dos atributos. A Figura 44 ilustra um exemplo de *ranking* para o conjunto de dados *flags*. Na figura, os números representam os índices dos atributos. Para o algoritmo BRSelectKBest(chi2), por exemplo, o primeiro atributo do *ranking* é o atributo com índice 18 desse conjunto de

dados.

Figura 44 – Exemplo do *ranking* resultante de alguns algoritmos de seleção de atributos multirrótulo para o conjunto de dados *flags*. Na figura, chi2 é a Estatística Qui-quadrado, F é o F-value da ANOVA e IM é a Informação Mútua.

Quanto ao número de atributos a serem selecionados, alguns trabalhos consideram esse número fixo, baseado em uma regra em função do número total de atributos (BOLÓN-CANEDO; SÁNCHEZ-MAROÑO; ALONSO-BETANZOS, 2012; KASHEF; NEZAMABADI-POUR; NIKPOUR, 2018). Neste trabalho, utilizamos um intervalo de valores para tornar a busca mais flexível, de modo que o número de atributos selecionados não fosse nem muito pequeno, levando a modelos subótimos, nem muito grande, treinando modelos semelhantes aos induzidos com todos os atributos, o que aumentaria o custo computacional do AutoML. Assim, propomos uma extensão da regra definida por Kashef, Nezamabadi-pour e Nikpour (2018), permitindo que o número de atributos a serem selecionados varie dentro de um intervalo previamente definido com base no número de atributos nos conjuntos de dados (d), conforme especificado no Algoritmo 2.

Algoritmo 2 Regra para o número de atributos a serem selecionados.

Require: Entrada d – o número de atributos
if $d < 100$ **then**
    selecione de 40% a 80% dos d atributos
else if $100 \leq d < 500$ **then**
    selecione de 30% a 60% dos d atributos
else if $500 \leq d < 1000$ **then**
    selecione de 20% a 40% dos d atributos
else if $d \geq 1000$ **then**
    selecione de 10% a 20% dos d atributos
end if

7.4 Detalhes de Implementação e *Setup* de Execução

Dez conjuntos de dados do repositório `MULAN` foram incluídos nos experimentos: *bibtex*, *birds*, *corel5k*, *emotions*, *enron*, *flags*, *genbase*, *medical*, *scene* e *yeast*. Esses conjuntos de dados foram pré-processados manualmente, conforme descreve a Seção 4.2 do Capítulo 4. Os algoritmos de seleção de atributos multirrótulo do hiperespaço (Seção 7.3) foram codificados na linguagem de programação `Python`, utilizando as bibliotecas `scikit-learn`

(PEDREGOSA et al., 2011) e `scikit-rebate` (URBANOWICZ et al., 2018). Os algoritmos do hiperespaço foram executados nos *3-folds* dos conjuntos de dados pré-processados a fim de produzir o *ranking* dos atributos, que é utilizado por AutoMLFS.

AutoMLFS foi implementado com o algoritmo OB com Processos Gaussianos disponível na biblioteca `scikit-optimize` (HEAD et al., 2024). AutoMLFS faz a seleção de atributos usando os *rankings* pré-calculados e induz modelos ECCs para o cálculo de Macro F-score. Utilizamos o algoritmo ECC, com hiperparâmetros padrão (J48 como classificador base), disponível no *wrapper* `MEKA`[1] do `scikit-multilearn` (SZYMANSKI; KAJDANOWICZ, 2019). O procedimento de otimização com OB ocorreu por até 1000 iterações. Caso não houvesse mudanças no valor do objetivo por 200 iterações consecutivas, ocorria a parada antecipada. Os valores dos hiperparâmetros da OB foram definidos a partir de testes iniciais, garantindo que houvesse convergência. A estratégia AutoMLFS foi executada cinco vezes para cada conjunto de dados.

7.5 Comparando AutoMLFS com os *Baselines*

Os *baselines* deste experimento foram modelos ECCs induzidos com diferentes conjuntos de dados e a ferramenta `auto-sklearn` (FEURER et al., 2015). Os modelos ECCs foram induzidos com os dados originais, com dados pré-processados e com dados resultantes de seleção de atributos multirrótulo. Os dados originais são os conjuntos de dados listados na Seção 7.4, porém, sem pré-processamento.

Os algoritmos de seleção de atributos multirrótulo foram os algoritmos da biblioteca `PyIT-MLFS` – uma biblioteca de código aberto em `Python` que implementa algoritmos de seleção de atributos multirrótulo baseados em filtros e fundamentados na teoria da informação (ESKANDARI, 2022). Até onde sabemos, `PyIT-MLFS` é a única biblioteca disponível livremente que implementa a seleção de atributos multirrótulo. Dada essa escassez de implementações disponíveis na literatura, é importante comparar a estratégia AutoMLFS com os algoritmos de seleção de atributos de `PyIT-MLFS`. Além disso, todos os algoritmos de `PyIT-MLFS` são baseados em entropia ou informação mútua, fornecendo um *baseline* relevante para comparação com a estratégia proposta. Isso permite avaliar nosso hiperespaço em relação a diferentes algoritmos de seleção de atributos.

Os algoritmos de `PyIT-MLFS` também permitem a criação de um *ranking* de atributos para posterior seleção e indução de modelos. Para cada algoritmo da biblioteca `PyIT-MLFS`, pré-calculamos os *rankings* utilizando os conjuntos de dados pré-processados. No entanto, limitamos o tempo para o cálculo dos *rankings* a 30 dias. Os algoritmos *PPT with Mutual Information* (PPT-MI), IGMF, SCLS, LRFS, LSMFS e MLSMFS finalizaram o cálculo do *ranking* no período definido e foram incluídos como *baseline* juntamente com o ECC.

[1] Dentre os algoritmos de classificação multirrótulo do `MEKA` está o algoritmo ECC, disponível via *wrapper* para a biblioteca `MULAN`.

Para esses *baselines* com seleção de atributos, definimos um *grid* com porcentagens fixas de atributos a serem selecionados, baseando-se na regra definida no Algoritmo 2. O *grid* foi definido para limitar o número de modelos ECCs a serem induzidos e para assegurar diferentes valores para o número de atributos. Assim, seja d o número total de atributos de um conjunto de dados, o número de atributos que compõem o *grid* corresponde às porcentagens definidas na Tabela 9. A seleção de atributos utiliza o *ranking* pré-calculado dos atributos e as porcentagens da Tabela 9 para criar um novo conjunto de dados, que é então utilizado para induzir um modelo ECC.

Tabela 9 – *Grid* do número de atributos a serem selecionados com base no número de atributos (d) do conjunto de dados.

Número de	**Porcentagem (%)**							
Atributos (d)	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**
$d < 100$				x	x	x	x	x
$100 \leq d < 500$			x	x	x	x		
$500 \leq d < 1000$		x	x	x				
$d \geq 1000$	x	x						

`Auto-sklearn` foi outro *baseline* do experimento. `Auto-sklearn` é uma ferramenta AutoML projetada para classificação monorrótulo e com suporte para a classificação multirrótulo. Seu hiperespaço é composto principalmente por classificadores e algoritmos de seleção de atributos monorrótulo. *Random Forest*, *Decision Trees* e algoritmos que treinam um classificador por rótulo são os classificadores multirrótulo do hiperespaço do `auto-sklearn`. Quanto aos seletores de atributos do hiperespaço, o único algoritmo multirrótulo do hiperespaço é SelectFromModel com ExtraTrees. Assim, como não existem estratégias AutoML específicas para seleção de atributos multirrótulo na literatura, comparar o AutoMLFS com outra estratégia AutoML, como o `auto-sklearn`, é um *baseline* importante para avaliar o impacto da exploração de um hiperespaço totalmente dedicado à seleção de atributos multirrótulo.

`Auto-sklearn` automatiza o pré-processamento de dados, o pré-processamento de atributos e a busca pelo modelo de classificação. Para a execução desse *baseline*, desabilitamos o seu módulo de pré-processamento automatizado e utilizamos os conjuntos de dados pré-processados manualmente e particionados em *3-folds*. Isso foi feito para manter os mesmos conjuntos de dados pré-processados em todo o experimento. O módulo de pré-processamento de atributos do `auto-sklearn` não foi desabilitado, pois nosso interesse é que `auto-sklearn` trate os atributos de um conjunto de dados antes da indução dos modelos de classificação multirrótulo. Assim, além de algoritmos de seleção de atributos, `auto-sklearn` também utiliza algoritmos de construção e de extração de atributos da engenharia de atributos.

A execução do `auto-sklearn` ocorreu nas partições de treino dos três *folds* de um conjunto de dados e, a partir das partições de teste, foi possível calcular o Macro F-score médio. Para execução, limitamos o tempo total de execução do `auto-sklearn` a um dia e o tempo de execução de um algoritmo de classificação multirrótulo a 30 minutos, conforme as especificações do manual do `auto-sklearn` (FEURER et al., 2015). Foram feitas cinco execuções do `auto-sklearn` para cada conjunto de dados. A Tabela 10 resume os algoritmos *baselines* utilizados no experimento, incluindo o algoritmo de seleção de atributos e o conjunto de dados utilizados por cada *baseline*.

Tabela 10 – Resumo dos algoritmos *baselines* do experimento.

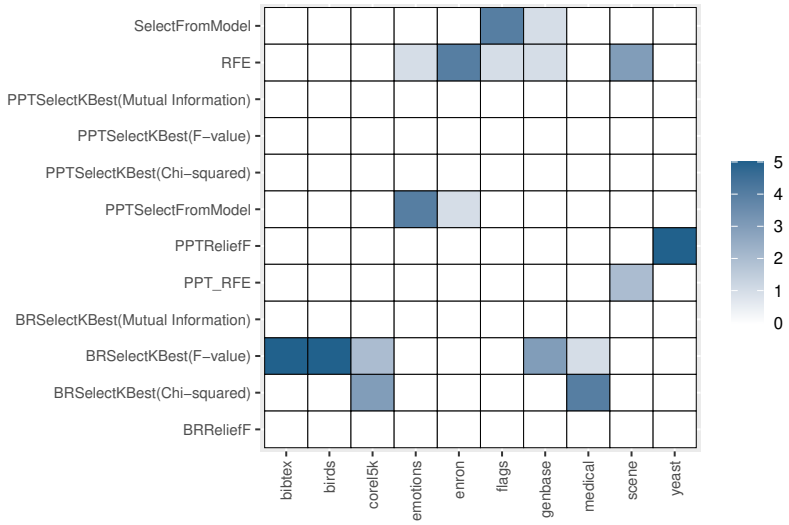
Algoritmo	**Algoritmo de Seleção de Atributos**	**Conjunto de Dados**
ECC	-	Original
ECC	-	Pré-processado
ECC	PPT-MI	Pré-processado
ECC	IGMF	Pré-processado
ECC	SCLS	Pré-processado
ECC	LRFS	Pré-processado
ECC	LSMFS	Pré-processado
ECC	MLSMFS	Pré-processado
`auto-sklearn`	-	Pré-processado

7.6 Resultados

O resultado de uma execução de AutoMLFS é um algoritmo de seleção de atributos e o número de atributos a ser selecionado que otimiza a função objetivo. Para apresentar e discutir os resultados de AutoMLFS quanto ao Macro F-score e ao número de atributos, adotamos como resultado das cinco execuções os seus valores médios. Para `auto-sklearn` não temos informação sobre o número de atributos selecionados, isso ocorre porque o hiperespaço do `auto-sklearn` contém diferentes algoritmos da engenharia de atributos, não necessariamente algoritmos de seleção de atributos. Portanto, os resultados para esse *baseline* foram apenas em relação ao Macro F-score médio.

Nomeamos os modelos ECCs induzidos com os conjuntos de dados originais (Or.) e pré-processados (PP.) como $ECC_{Or.}$ e $ECC_{PP.}$, respectivamente. Para esses *baselines*, tomamos como resultados os valores médios resultantes de cinco execuções. Quando utilizamos um algoritmo de seleção de atributos (*Feature Selection* (FS)), denominamos o método como ECC_{FS} (ECC_{IGMF}, por exemplo). Para os *baselines* que usam seleção de atributos, diferentes valores para o número de atributos foram adotados, de acordo com a regra definida na Tabela 9. Assumimos como resultado desses *baselines* o número de atributos que propiciou o modelo ECC com o melhor desempenho (Macro F-score).

Figura 45 – Algoritmos de seleção de atributos multirrótulo recomendados por AutoMLFS.



AutoMLFS recomenda diferentes algoritmos de seleção de atributos e números de atributos a serem selecionados a depender da execução e do conjunto de dados. A Figura 45 ilustra o número de vezes que cada algoritmo de seleção de atributos do hiperespaço foi escolhido por AutoMLFS. As versões do algoritmo SelectKBest com transformação PPT, o algoritmo BRSelectKBest com método de Informação Mútua e o algoritmo BRReliefF não apareceram nos resultados da estratégia automatizada para nenhum dos conjuntos de dados. Os demais algoritmos de seleção de atributos estão nos resultados de pelo menos um conjunto de dados. É possível observar que diferentes algoritmos de seleção de atributos foram recomendados a depender do conjunto de dados ou mesmo da execução, que é influenciada por fatores aleatórios. Os resultados referentes ao número de atributos, Macro F-score, testes estatísticos e tempo de execução são descritos a seguir, nas Seções 7.6.1 a 7.6.4.

7.6.1 Número de Atributos

A Figura 46 ilustra a quantidade de atributos utilizados pelos algoritmos avaliados. Os *baselines* induzidos com os conjuntos de dados originais e pré-processados utilizaram todos os atributos disponíveis. Já AutoMLFS e os *baselines* com seleção de atributos utilizaram números de atributos menores. No conjunto de dados *corel5k*, por exemplo, o número de atributos nos conjuntos de dados original e pré-processado era 499. A estratégia AutoMLFS encontrou soluções com número de atributos entre 149 e 299 e os *baselines* trabalharam com número de atributos iguais a 149, 199, 249 e 299, para esse conjunto de dados.

Para ajudar a visualizar as diferenças entre o número de atributos selecionados, a

Figura 46 – Número de atributos selecionados/utilizados pelos algoritmos.

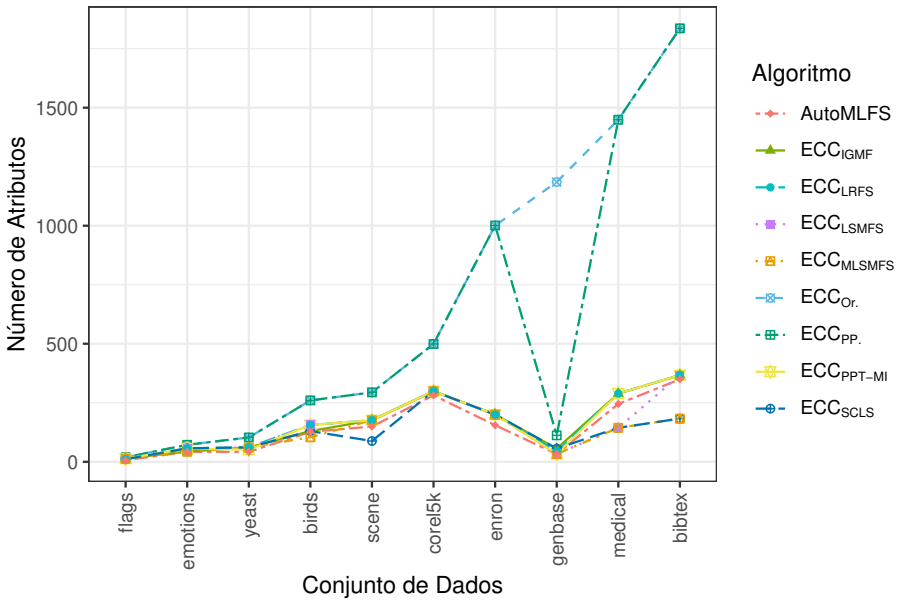


Tabela 11 mostra a quantidade de atributos correspondente aos valores representados na Figura 46. A partir da tabela, podemos ver que AutoMLFS reduziu significativamente a dimensionalidade dos dados na maioria dos conjuntos de dados, recomendando o menor número de atributos em seis dos dez conjuntos analisados. Essa forte redução trouxe melhorias significativas nos valores de Macro F-score em muitos casos, enquanto em outros, o desempenho em Macro F-score manteve-se competitivo.

Tabela 11 – Número de atributos selecionados/utilizados pelos algoritmos. Os menores (melhores) valores são destacados com sublinhado.

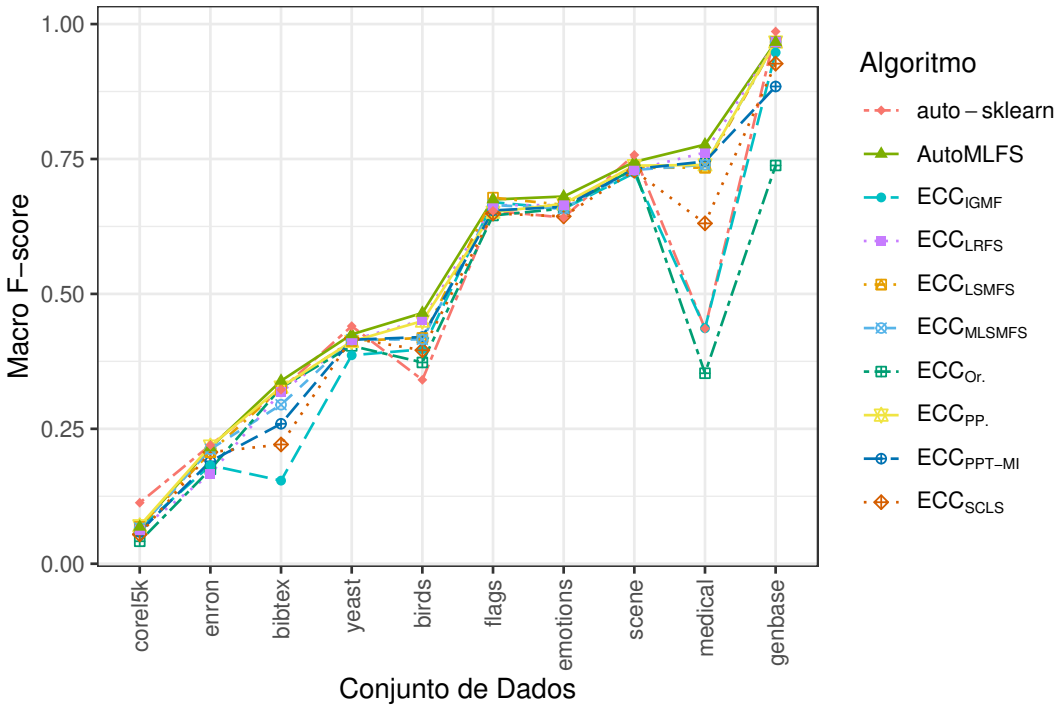
Conjunto de Dados	**AutoMLFS**	**ECC**							
		Or.	**PP.**	**PPT-MI**	**IGMF**	**SCLS**	**LSMFS**	**MLSMFS**	**LRFS**
bibtex	349 (02)	1836	1836	367	367	183	367	183	367
birds	127 (00)	260	260	78	130	130	156	104	156
corel5k	282 (10)	499	499	299	299	299	299	299	299
emotions	42 (2)	72	72	57	43	57	57	43	57
enron	155(26)	1001	1001	200	200	200	200	200	200
flags	7(00)	19	19	13	15	11	15	13	11
genbase	33 (00)	1185	112	44	56	56	33	33	44
medical	245(2.387)	1449	1449	289	289	144	144	144	289
scene	149(24)	294	294	176	176	88	176	176	176
yeast	42(00)	103	103	51	61	61	61	61	61

7.6.2 Macro F-score

Em relação ao Macro F-score, a Figura 47 mostra os valores médios dessa medida para AutoMLFS, `auto-sklearn`, $ECC_{Or.}$ e $ECC_{PP.}$, além dos melhores valores resultantes dos modelos ECCs treinados com os algoritmos da biblioteca `PyIT-MLFS`. $ECC_{Or.}$

têm desempenho inferior ao baseline $ECC_{PP.}$. Nesse caso, as maiores diferenças de desempenho aconteceram em conjuntos de dados que possuem pequena incidência de alguns rótulos, o que prejudica o aprendizado, uma vez que não há dados suficientes para mapear dados de entrada em dados de saída (rótulos). AutoMLFS tem Macro F-score médio igual ou superior à maioria dos *baselines*. Considerando a redução significativa no número de atributos proporcionada por AutoMLFS (Figura 46) e, consequentemente, no tamanho dos modelos ECC, concluímos que nossa proposta melhorou com sucesso os resultados de classificação. Embora as linhas no gráfico se sobreponham, é possível notar melhorias de desempenho em diferentes conjuntos de dados, como *birds*, *emotions* e *medical*.

Figura 47 – Macro F-score dos algoritmos avaliados.



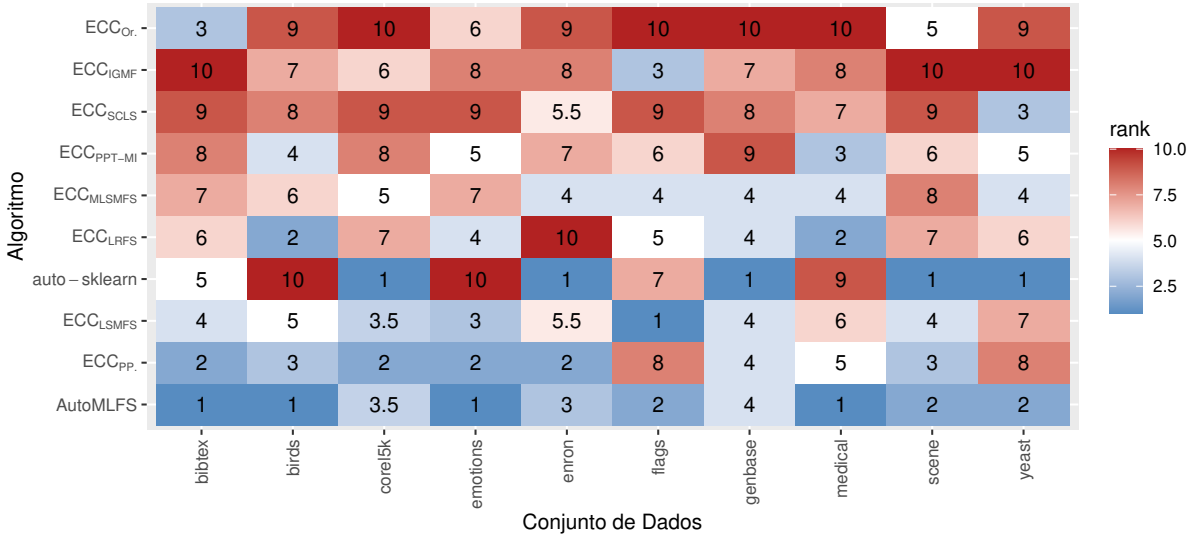
Para ajudar a visualizar os resultados de Macro F-score, a Tabela 12 apresenta os valores numéricos mostrados na Figura 47. Podemos observar na tabela que, na grande maioria dos conjuntos de dados, o desempenho do AutoMLFS em Macro F-score foi superior ou competitivo em comparação aos outros algoritmos *baselines*. Ao comparar especificamente AutoMLFS com `auto-sklearn`, há casos em que o AutoMLFS obteve melhorias significativas, como nos conjuntos de dados: *birds* (0.4647 vs. 0.3408) e *medical* (0.7767 vs. 0.4357). Por outro lado, o `auto-sklearn` nunca obteve uma vantagem tão expressiva contra AutoMLFS. Além disso, ao analisar os valores de Macro F-score em conjunto com o número de atributos selecionados, novamente não observamos perdas significativas no desempenho do AutoMLFS em relação aos algoritmos *baselines*. Pelo contrário, há ganhos de desempenho e resultados altamente competitivos, mesmo com

uma redução significativa na dimensionalidade dos dados.

Tabela 12 – Valores médios do Macro F-score para AutoMLFS, `auto-sklearn`, $ECC_{Or.}$ e $ECC_{PP.}$, e valores do Macro F-score para ECC com seletores de atributos da biblioteca `PyIT-MLFS`. Os melhores resultados estão destacados com um sublinhado.

Conj. Dados	AutoMLFS	auto-sklearn	ECC							
			Or.	PP.	PPT-MI	IGMF	SCLS	LSMFS	MLSMFS	LRFS
bibtex	0.339(0.000)	0.322(0.011)	0.326	0.327	0.259	0.154	0.221	0.325	0.295	0.318
birds	0.465(0.000)	0.341(0.026)	0.373	0.450	0.420	0.397	0.396	0.418	0.416	0.451
corel5k	0.068(0.000)	0.113(0.002)	0.042	0.070	0.059	0.066	0.054	0.068	0.066	0.063
emotions	0.68(0.003)	0.641(0.006)	0.658	0.667	0.661	0.658	0.644	0.665	0.658	0.665
enron	0.215(0.001)	0.220(0.004)	0.175	0.218	0.190	0.182	0.207	0.207	0.212	0.166
flags	0.675(0.000)	0.654(0.001)	0.645	0.651	0.655	0.670	0.649	0.678	0.665	0.664
genbase	0.967(0.000)	0.986(0.000)	0.738	0.967	0.884	0.947	0.927	0.967	0.967	0.967
medical	0.777(0.001)	0.436(0.009)	0.353	0.739	0.745	0.436	0.631	0.734	0.739	0.762
scene	0.745(0.001)	0.758(0.011)	0.733	0.738	0.731	0.724	0.728	0.734	0.730	0.730
yeast	0.425(0.000)	0.440(0.006)	0.404	0.412	0.415	0.386	0.418	0.413	0.415	0.414

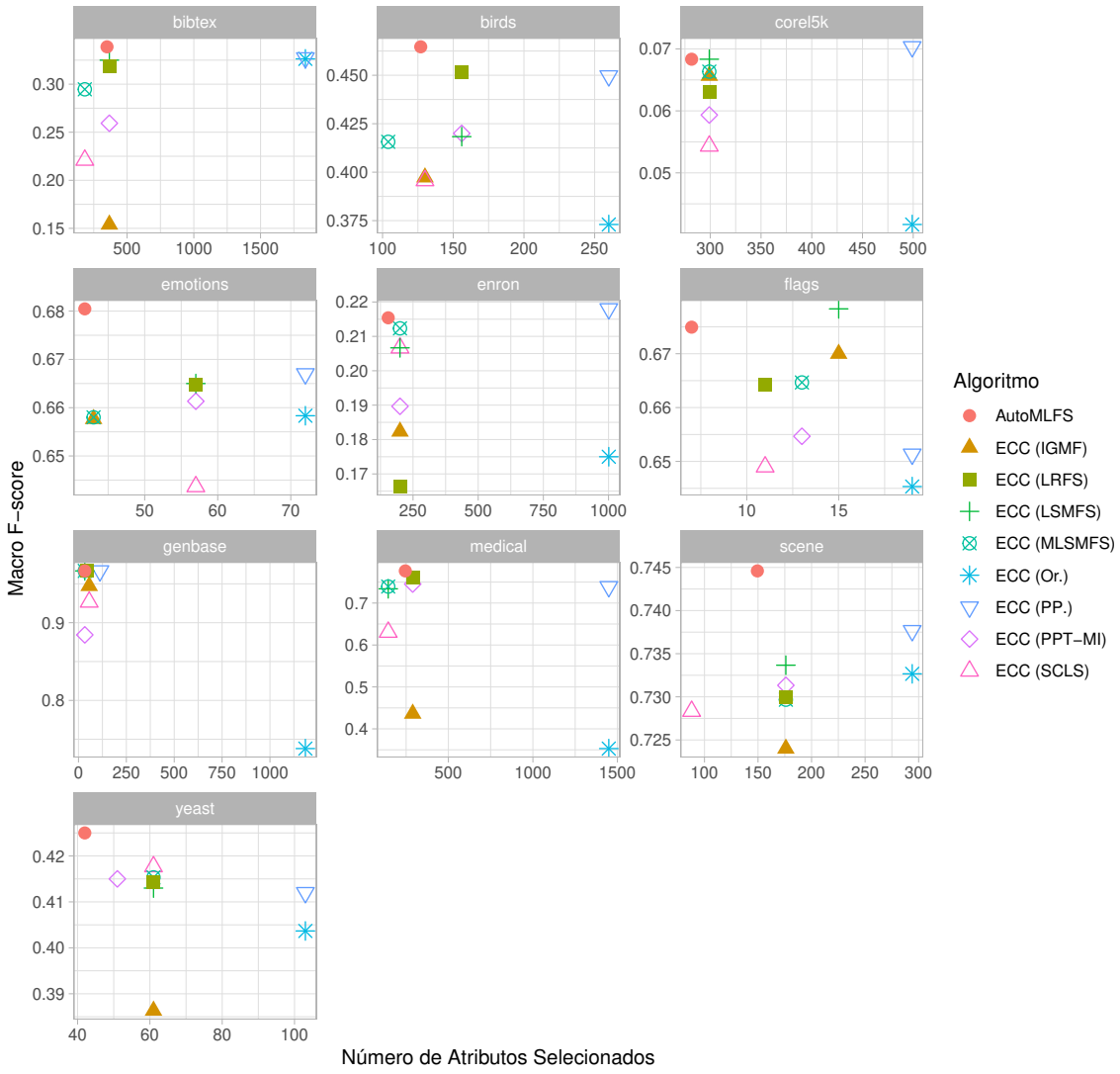
Figura 48 – *Ranking* dos algoritmos avaliados no experimento em relação ao Macro F-score. Na figura, *rankings* iguais para os algoritmos de um conjunto de dados representam algoritmos que tiveram Macro F-scores iguais. Os melhores algoritmos têm *rankings* menores.



Na Figura 48, os algoritmos foram classificados em relação ao Macro F-score. Os *rankings* menores representam os algoritmos com melhores Macro F-scores (quadrados azuis), enquanto os *rankings* maiores representam os algoritmos com piores desempenhos (quadrados vermelhos). Na Figura, os algoritmos estão dispostos pelo *ranking* médio. AutoMLFS foi classificado entre os três melhores algoritmos em todos os conjuntos de

dados, alcançando a melhor média de *ranking* entre os algoritmos avaliados. Isso indica que, em média, a estratégia proposta obteve o melhor desempenho, seguida por $ECC_{PP.}$, ECC_{LSMFS} e `auto-sklearn`. Embora `auto-sklearn` tenha obtido os melhores valores de Macro F-score em cinco dos dez conjuntos de dados avaliados, seu desempenho nos conjuntos *birds*, *emotions*, *medical* e *flags* ficou entre os piores. Essa inconsistência no desempenho do `auto-sklearn` é uma desvantagem significativa ao lidar com cenários multirrótulo. Por outro lado, AutoMLFS apresentou resultados altamente consistentes em todos os conjuntos de dados, provando ser uma excelente alternativa para seleção de atributos multirrótulo.

Figura 49 – Relacionamento entre o Macro F-score e o número de atributos selecionados.



7.6.3 Visão Geral

A Figura 49 apresenta uma visualização do Macro F-score e do número de atributos resultantes dos algoritmos avaliados. Conforme mencionamos, não foi possível realizar essa análise com `auto-sklearn`, pois esta estratégia não disponibiliza o número de atributos selecionados. A partir da figura, podemos confirmar que AutoMLFS superou algoritmos de seleção de atributos multirrótulo em seis dos conjuntos de dados analisados, mantendo desempenho competitivo nos demais casos avaliados. Considerando a relação entre Macro F-score e número de atributos, os gráficos da Figura 49 podem ser subdivididos em quatro quadrantes. O primeiro quadrante é a parte superior direita do gráfico. O segundo quadrante é a parte superior esquerda. O terceiro e o quarto quadrantes estão à esquerda e à direita na parte inferior do gráfico, respectivamente.

Na maioria dos conjuntos de dados, $ECC_{PP.}$ está no primeiro quadrante, sendo um algoritmo com Macro F-score alto e muitos atributos. No segundo quadrante estão os algoritmos com alto desempenho e poucos atributos. Podemos ver que AutoMLFS está no segundo quadrante para todos os conjuntos de dados, provando ser uma excelente estratégia AutoML para obter alta performance com um número menor de atributos. Os algoritmos com desempenho e número de atributos baixos estão no terceiro quadrante. ECC_{IGMF} e ECC_{SCLS} são os algoritmos que frequentemente aparecem nesse quadrante, demonstrando ser boas opções para a redução de dimensionalidade, mas sem manter bons desempenhos preditivos. No quarto quadrante, os algoritmos têm baixo desempenho e um elevado número de atributos. $ECC_{Or.}$ está no quarto quadrante na maioria dos conjuntos de dados.

Para confirmar os resultados discutidos até o momento, aplicamos o teste de Friedman para verificar se os algoritmos avaliados apresentavam desempenhos estatisticamente significativos em termos do Macro F-score. A hipótese nula foi rejeitada no teste de Friedman e o teste *post-hoc* de Nemenyi foi então aplicado para encontrar os grupos de algoritmos com desempenhos estatisticamente diferentes. A Figura 50 apresenta o diagrama de diferença crítica do teste de Nemenyi. Algoritmos conectados no diagrama de diferença crítica são aqueles para os quais não foram encontradas diferenças estatisticamente significativas.

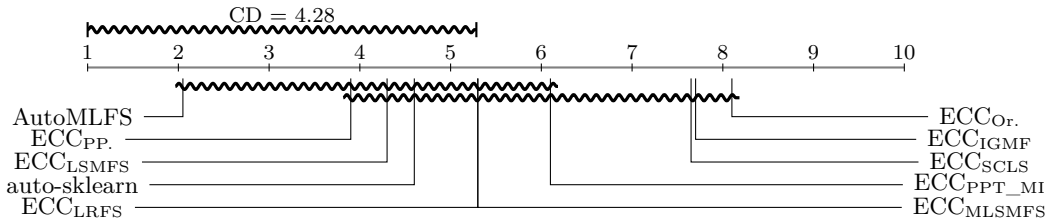


Figura 50 – Diagrama de diferença crítica dos algoritmos avaliados.

De acordo com a Figura 50, AutoMLFS obteve a primeira posição no *ranking* médio dos algoritmos, comprovando que ele alcançou o maior Macro F-score médio. Na sequência, os algoritmos com melhores Macro F-scores foram $ECC_{PP.}$, ECC_{LSMFS}, `auto-sklearn`,

ECC_{LRFS}, ECC_{MLSMFS} e ECC_{PPT-MI}. Embora AutoMLFS não tenha sido estatisticamente superior a alguns algoritmos *baselines*, o gráfico mostra uma grande diferença de classificação entre AutoMLFS e o segundo colocado ($ECC_{PP.}$), atestando a superioridade da nossa proposta. O gráfico também demonstra que, em média, nenhum dos algoritmos *baselines* superou $ECC_{PP.}$ (que é o ECC executado sem seleção de atributos). Mesmo `auto-sklearn`, que explora um hiperespaço de algoritmos da engenharia de atributos, não conseguiu superar os resultados médios obtidos usando todos os atributos. Isso prova que o AutoMLFS foi o único algoritmo capaz de realmente melhorar os resultados de desempenho enquanto reduzia a dimensionalidade dos atributos, o que pode posicioná-lo como o novo algoritmo estado da arte em AutoML para seleção de atributos multirrótulo.

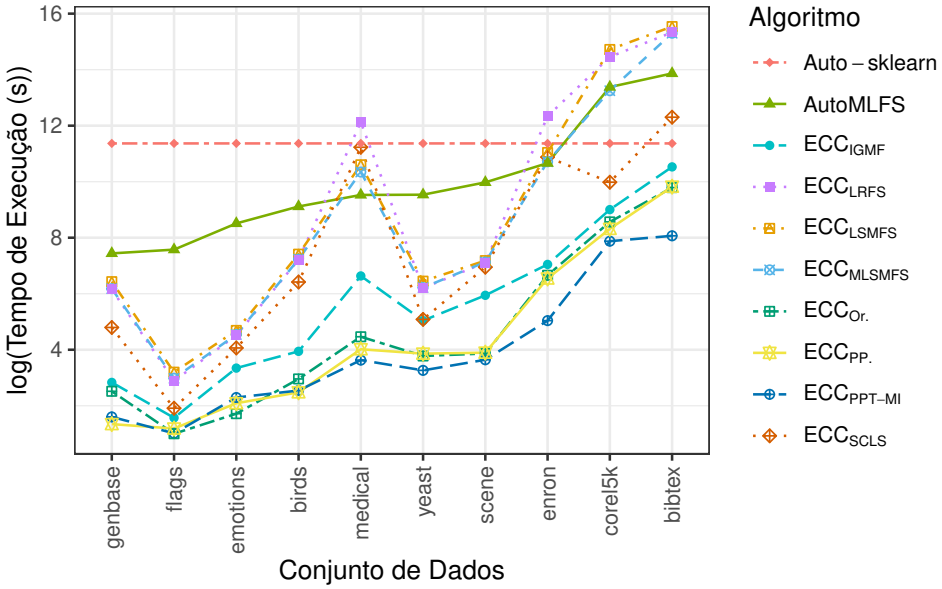
A superioridade de AutoMLFS pode ser justificada pela exploração do hiperespaço, que avalia diferentes algoritmos de seleção de atributos e número de atributos em busca daquele com o melhor desempenho. Assim, AutoMLFS induz modelos com número de atributos reduzido e com desempenho superior. De forma similar, a exploração do hiperespaço do `auto-sklearn` também contribui para o seu bom desempenho, uma vez que `auto-sklearn` encontra a melhor solução testando diferentes algoritmos de classificação multirrótulo e da engenharia de atributos. Não podemos fazer afirmações sobre o número de atributos das soluções (ou modelos) encontrados pelo `auto-sklearn`. No entanto, esse número pode variar de valores muito pequenos a grandes, pois o algoritmo de engenharia de atributos pode reduzir ou aumentar a dimensionalidade dos dados.

O desempenho de ECC_{LSMFS} e de outros algoritmos *baselines* com seleção de atributos pode ser justificado pela qualidade do *ranking* resultante de cada algoritmo de seleção de atributos e pelo número de atributos utilizados para induzir os modelos ECCs. $ECC_{PP.}$ foi induzido com dados pré-processados e está no grupo de algoritmos com os melhores desempenhos. $ECC_{Or.}$ foi induzido com dados originais e está no grupo de algoritmos com os piores desempenhos. Isso demonstra que o pré-processamento de dados contribui para a indução de modelos mais precisos.

7.6.4 Uma Nota sobre o Tempo de Execução

Por fim, gostaríamos de fazer algumas observações sobre os tempos de execução dos algoritmos. A Figura 51 descreve seus valores em uma escala logarítmica. Como o AutoMLFS é uma estratégia baseada em otimização, sua execução considera todo o processo de otimização. Assim, seu tempo médio de execução foi estimado como a média das cinco execuções independentes, mais o tempo necessário para calcular o *rank* dos atributos de cada algoritmo de seleção de atributos. O outro *baseline* AutoML (`auto-sklearn`) não possui opções de alto nível para controlar sua execução, apenas um tempo de execução regular, que foi definido como um dia para cada algoritmo de seleção e conjunto de dados. Vale ressaltar que o AutoMLFS foi mais rápido e preciso (Macro F-score) do que o `auto-sklearn` na maioria dos conjuntos de dados.

Figura 51 – Análise do tempo de execução dos algoritmos avaliados no experimento.



Os *baselines* $ECC_{Or.}$ e $ECC_{PP.}$ também expressaram seu tempo de execução pela média de cinco repetições. O tempo de execução dos demais *baselines* considerou o tempo para estimar o *rank* dos atributos mais o tempo de execução do ECC, considerando o melhor resultado. O único resultado não intuitivo é que o ECC configurado com alguns algoritmos de seleção de atributos foi computacionalmente mais custoso do que AutoMLFS para alguns conjuntos de dados. Isso ocorreu devido à complexidade de alguns algoritmos de seleção de atributos, que é potencializada pela alta dimensionalidade dos conjuntos de dados. Em contraste, o hiperespaço de busca de AutoMLFS inclui algoritmos de seleção de atributos menos complexos.

7.7 Considerações Finais

Neste capítulo propomos AutoMLFS, uma estratégia para seleção automática de atributos. O hiperespaço continha um conjunto de algoritmos de seleção de atributos multirrótulo e os possíveis números de atributos para seleção. A OB com Processos Gaussianos foi responsável por buscar o algoritmo de seleção de atributos multirrótulo e o número de atributos para a indução de um modelo ECC com Macro F-score máximo. Para confirmar a superioridade da estratégia proposta, nós comparamos AutoMLFS com os *baselines* `auto-sklearn` e com modelos ECCs induzidos com conjuntos de dados nos formatos original, pré-processado e resultantes da seleção de atributos. PPT-MI, IGMF, SCLS, LSMFS, MLSMFS e LRFS foram os algoritmos de seleção de atributos multirrótulo dos *baselines*. Esses algoritmos estão disponíveis na biblioteca `PyIT-MLFS`.

O pré-processamento manual dos dados melhorou o desempenho dos modelos

ECC. Esse fato foi observado ao comparar $ECC_{Or.}$ e $ECC_{PP.}$. Além disso, os experimentos comprovaram que AutoMLFS alcançou os melhores valores médios para o Macro F-score. Estatisticamente, o desempenho de AutoMLFS foi similar ao desempenho dos algoritmos *baselines* $ECC_{PP.}$, ECC_{LSMFS}, `auto-sklearn`, ECC_{LRFS}, ECC_{MLSMFS} e ECC_{PPT-MI}. Contudo, o diagrama de diferença crítica de Nemenyi (Figura 50) mostrou uma grande distância na classificação entre AutoMLFS e o segundo colocado ($ECC_{PP.}$), assim como em relação aos outros métodos de referência, atestando a superioridade de nossa proposta.

AutoMLFS e os algoritmos *baseline* de seleção de atributos recomendaram um número reduzido de atributos em comparação com $ECC_{Or.}$ e $ECC_{PP.}$. A redução da dimensionalidade dos conjuntos de dados acelerou o treinamento dos modelos ECC e pode ter melhorado a interpretabilidade desses modelos. Não foi possível obter essas conclusões para `auto-sklearn`, pois sua implementação não fornece informações sobre a quantidade de atributos selecionados.

Tanto AutoMLFS quanto os algoritmos *baselines* de seleção de atributos fornecem uma classificação dos atributos selecionados. Embora a complexidade do cálculo dessa classificação varie conforme cada algoritmo de seleção, as estratégias AutoML apresentam uma maior complexidade geral do que os algoritmos *baselines*. Enquanto AutoMLFS e `auto-sklearn` induzem e avaliam diversos modelos de classificação multirrótulo, utilizando vários algoritmos de seleção de atributos, cada algoritmo *baseline* avalia apenas um único modelo. Consequentemente, a complexidade de AutoMLFS é superior à dos algoritmos *baselines*. No entanto, essa complexidade maior é compensada por sua capacidade de recomendar os algoritmos e atributos que proporcionam o melhor desempenho global.

Logo, confirmamos nossa hipótese de pesquisa. AutoMLFS encontra automaticamente o algoritmo de seleção de atributos multirrótulo e o número de atributos que permite a indução de modelos precisos, utilizando um número de atributos reduzido. No próximo capítulo, estamos interessados em integrar a seleção e atributos multirrótulo à estratégia EMANUEL. Assim, os resultados obtidos neste capítulo, bem como a definição do hiperespaço de AutoMLFS, serviram de embasamento para a extensão de EMANUEL no Capítulo 8.

Capítulo 8

EMANUEL$_{\mathbf{FS}}$

No cenário multirrótulo, o AutoML foi aplicado especificamente na busca do melhor classificador multirrótulo, considerando um objetivo de otimização (SÁ; PAPPA; FREITAS, 2017; SÁ; FREITAS; PAPPA, 2018; SÁ et al., 2020; WEVER et al., 2021). Nesta tese, estendemos esse cenário e trabalhamos com a busca pelo melhor classificador multirrótulo utilizando a otimização multiobjetivo, propondo EMANUEL. No entanto, as demais tarefas do *pipeline* de AM como as relacionadas ao pré-processamento dos dados e à engenharia de atributos, também poderiam ser automatizadas.

Este capítulo investiga a viabilidade da automatização da tarefa engenharia de atributos do *pipeline*, mais especificamente da seleção de atributos, juntamente com a busca pelo classificador multirrótulo. Para isso, estendemos a estratégia EMANUEL (Capítulo 5), de forma que ela automatize tanto a seleção de atributos quanto a busca pelo classificador multirrótulo. Denominamos a nova estratégia como *gEnetic Multi-objective strategy for the Automatic selectioN of mUlti-labEl cLassifiers and Feature Selection algorithms* (EMANUEL$_{FS}$).

De acordo com Kashef, Nezamabadi-pour e Nikpour (2018), a seleção de atributos tem como objetivo identificar, dentre os atributos disponíveis, um subconjunto de atributos relevantes, preservando as características principais dos dados, de forma a manter ou melhorar a eficácia dos algoritmos de AM. Dessa forma, nossa hipótese de pesquisa pressupõe que a estratégia EMANUEL$_{FS}$ pode encontrar combinações ótimas de algoritmos de seleção de atributos e de classificação multirrótulo que preservem ou até melhorem o desempenho de EMANUEL.

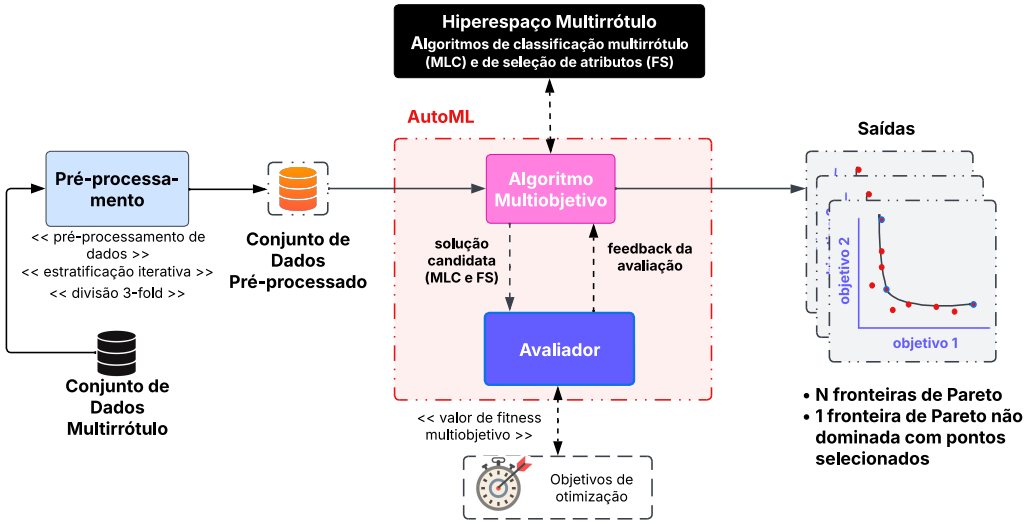
Este capítulo está organizado da seguinte forma: a Seção 8.1 explica o funcionamento de EMANUEL$_{FS}$; a Seção 8.2 delimita os algoritmos e hiperparâmetros do hiperespaço de EMANUEL$_{FS}$; a Seção 8.3 descreve a representação dos indivíduos; a Seção 8.4 apresenta

os detalhes de implementação da estratégia proposta; a Seção 8.4 identifica os *baselines* de comparação e o *setup* de execução; a Seção 8.6 apresenta e discute os resultados; e a Seção 8.7 traz as considerações finais.

8.1 Como EMANUEL$_{FS}$ Trabalha?

O funcionamento de EMANUEL$_{FS}$ é ilustrado na Figura 52. A estratégia EMANUEL$_{FS}$ tem como entrada um conjunto de dados multirrótulo. O fluxo de trabalho inicia-se com o pré-processamento manual dos dados. Esse conjunto é submetido ao AutoML, onde um algoritmo de otimização multiobjetivo é responsável por selecionar no hiperespaço uma solução candidata – composta pelo algoritmo de seleção de atributos, pelo algoritmo de classificação e pelas respectivas configurações de hiperparâmetros – e enviar a solução candidata ao avaliador. Neste experimento, adotamos o algoritmo de otimização multiobjetivo NSGA-II. O NSGA-II trabalha com uma população de indivíduos, sendo que cada indivíduo é uma codificação de um algoritmo de seleção de atributos e de um algoritmo de classificação, ambos multirrótulo. A decodificação do indivíduo em uma solução candidata é feita com base no hiperespaço, que determina quais algoritmos e configurações de hiperparâmetros um indivíduo pode assumir.

Figura 52 – Fluxograma do funcionamento da estratégia EMANUEL$_{FS}$.



O avaliador executa o algoritmo de seleção de atributos e obtém um novo conjunto de dados com o número de atributos reduzido. Em seguida, o avaliador induz um modelo de classificação multirrótulo, utilizando o novo conjunto de dados. O avaliador avalia o modelo induzido, calculando os valores dos objetivos e retornando um *feedback* com as avaliações ao algoritmo de otimização. O processo de otimização continua até que uma

condição de parada seja alcançada. Neste experimento, os objetivos de otimização foram o Macro F-score e o tamanho do modelo.

O resultado da estratégia EMANUEL$_{FS}$ é uma fronteira de Pareto. Para N execuções da estratégia EMANUEL$_{FS}$, haverá N fronteiras de Pareto. Utilizamos o conceito de não dominância apresentado por Deb et al. (2002) para encontrar a primeira fronteira de Pareto não dominada. Essa fronteira contém várias soluções. No entanto, não exploramos todas as soluções da fronteira como resultado de EMANUEL$_{FS}$. Utilizamos o conceito de *Frugality Score*, descrito na Seção 5.1.4, para selecionar três soluções da fronteira como resultados de EMANUEL$_{FS}$. Para maior embasamento do funcionamento de EMANUEL$_{FS}$, as Seções 8.2 e 8.3 descrevem o hiperespaço e a representação dos indivíduos do NSGA-II, respectivamente.

8.2 Hiperespaço

Durante a otimização, o NSGA-II avalia de forma conjunta diferentes algoritmos de seleção de atributos e de classificação multirrótulo. O hiperespaço especifica quais são esses algoritmos e também quais são os possíveis valores de cada hiperparâmetro. O hiperespaço de EMANUEL$_{FS}$ é composto por 18 opções para a seleção de atributos, 21 algoritmos de classificação multirrótulo e 18 algoritmos de classificação monorrótulo.

O hiperespaço de seleção de atributos multirrótulo possui duas opções: 1) não realizar a seleção de atributos ou 2) realizar a seleção utilizando um dos 17 algoritmos disponíveis. Se a opção escolhida for não realizar a seleção de atributos, o conjunto de dados com todos os atributos é utilizado para a indução do modelo de classificação. Caso contrário, um dos 17 algoritmos é selecionado, sendo então realizada a seleção de atributos antes da indução do modelo de classificação. A lista completa dos algoritmos de seleção de atributos multirrótulo do hiperespaço inclui: D2F, IGMF, LRFS, LSMFS, *Max-dependency and Min-redundancy* (MDMR), MLSMFS, *Pairwise Multi-label Utility* (PMU), PPT-MI, SCLS, SelectFromModel, RFE, BRSelectKBest, PPTSelectKBest, BRReliefF, PPTReliefF, PPT_RFE e PPTSelectFromModel.

Em relação aos hiperparâmetros, o hiperespaço utiliza um conjunto finito de valores discretizados. Essa abordagem de discretização dos hiperparâmetros já foi utilizada no hiperespaço de classificação multirrótulo das estratégias EMANUEL e EMANUEL$_{SM}$ (Capítulos 5 e 6). Para o hiperparâmetro "número de atributos a serem selecionados", os possíveis valores desse hiperparâmetro são definidos de acordo com o Algoritmo 2 do Capítulo 7. Para mais informações sobre o hiperespaço de seleção de atributos, consulte o Apêndice F.

O hiperespaço de classificação multirrótulo de EMANUEL$_{FS}$ expande o hiperespaço da estratégia EMANUEL, incorporando os três novos algoritmos: MLkNN, *Hierarchy Of Multi-labEl leaRners* (HOMER) e ECC. Já o hiperespaço de classificação monorrótulo de

EMANUEL$_{FS}$ é igual ao hiperespaço da estratégia EMANUEL. Para detalhes adicionais sobre o hiperespaço de classificação monorrótulo/multirrótulo, consulte o Apêndice A.

8.3 Indivíduo

O indivíduo de EMANUEL$_{FS}$ é uma extensão do indivíduo de EMANUEL. A diferença é que o indivíduo passa a representar também o algoritmo de seleção de atributos. Ele é uma codificação de números inteiros, representado por meio de um vetor, onde cada posição corresponde a um gene. Os genes são organizados da seguinte forma:

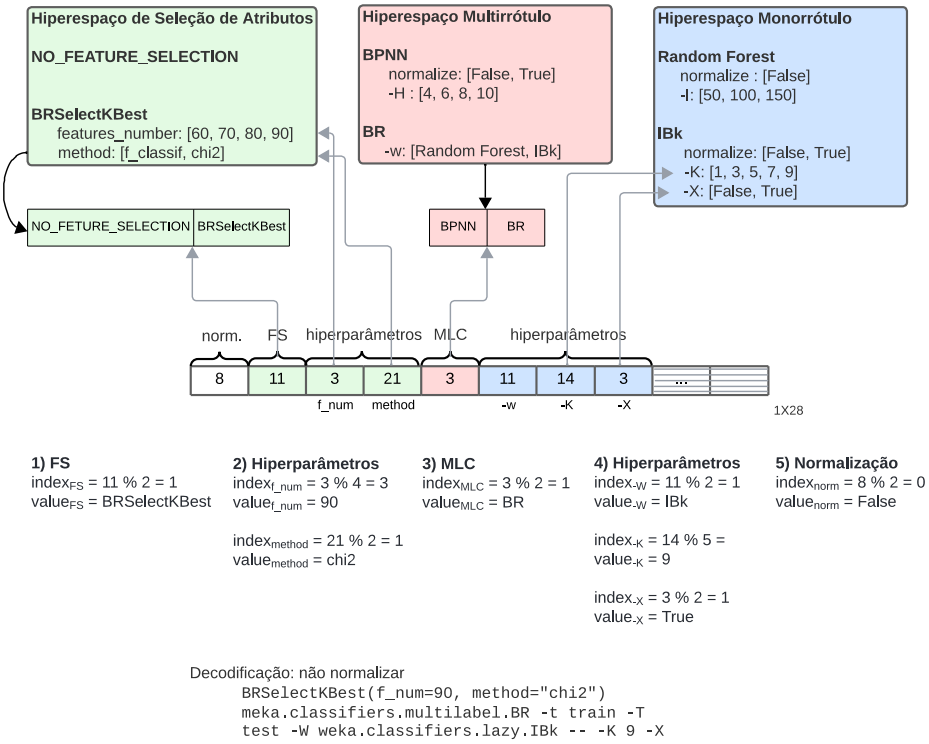
- O primeiro gene indica se haverá ou não normalização;
- O segundo gene define o algoritmo de seleção de atributos;
- Os genes seguintes representam os hiperparâmetros do algoritmo de seleção;
- Após os hiperparâmetros, o gene representa o algoritmo de classificação multirrótulo;
- Os genes restantes representam os hiperparâmetros do algoritmo de classificação multirrótulo.

O vetor numérico é decodificado primeiramente em um algoritmo de seleção de atributos e, em seguida, em um algoritmo de classificação multirrótulo. A normalização é decodificada por último, pois ela depende do algoritmo de classificação. As fórmulas para a decodificação são as mesmas utilizadas em EMANUEL. Elas são definidas nas Equações 11 e 12 da Seção 5.1.3. O indivíduo armazena números que são utilizados para indexar vetores do hiperespaço. Portanto, a Equação 11 determina o índice de um vetor do hiperespaço (`index`), calculando o resto da divisão do valor do gene (`gene_value`) pelo tamanho do vetor do hiperespaço (`array_values`). Já a Equação 12 recupera um algoritmo ou o valor de um hiperparâmetro, consultando o vetor do hiperespaço (`array_values`) no índice (`index`) resultante da Equação 11.

A Figura 53 ilustra um exemplo de decodificação de um indivíduo de EMANUEL$_{FS}$ para fins didáticos. O hiperespaço da figura possui apenas dois algoritmos de seleção de atributos e dois algoritmos de classificação multirrótulo. A decodificação começa com o algoritmo de seleção de atributos. É possível obter do hiperespaço um vetor com os algoritmos de seleção de atributos. Neste caso, `array_values = [NO_FEATURE_SELECTION, BRSelectKBest]`. O valor do gene para o algoritmo de seleção de atributos é 11 (`gene_value = 11`). Utilizando a Equação 11, o resto da divisão de `gene_value` pelo tamanho de `array_values` é um. Logo `index = 1`. Pela Equação 12, `value` é o algoritmo `BRSelectKBest` (`value = array_value[1]`).

A decodificação continua com os hiperparâmetros do algoritmo `BRSelectKBest` no hiperespaço de seleção de atributos. Os hiperparâmetros desse algoritmo são

Figura 53 – Exemplo ilustrativo da decodificação de um indivíduo de EMANUEL$_{FS}$. Na figura, norm. é a normalização, MLC (*Multi-label Classification*) é o algoritmo de classificação multirrótulo e FS (*Feature Selection*) é o algoritmo de seleção de atributos.



`method` e `features_number`. Para `features_number`, há quatro valores no hiperespaço (`array_values = [60, 70, 80, 90]`), e o valor do terceiro gene é três (`gene_value = 3`). O resultado da Equação 11 é `index = 3%4 = 3` e da Equação 12 é `value = array_values[3] = 90`. O hiperparâmetro `method` pode assumir os valores `f_classif` ou `chi2`. O valor do quarto gene é 21 (`gene_value = 21`). O resultado da Equação 11 é `index = 21%2 = 1` e da Equação 12 é `value = array_values[1] = chi2`. Portanto, o algoritmo de seleção de atributos é `BRSelectKBest` e seus hiperparâmetros são `features_number = 90` e `method = chi2`. Os próximos passos envolvem a decodificação do algoritmo de classificação multirrótulo e da normalização. Entretanto, não descreveremos o restante da decodificação aqui, pois a Figura 14 da Seção 5.1.3 já apresentou o mesmo exemplo anteriormente.

8.4 Detalhes de Implementação

Os algoritmos de seleção de atributos multirrótulo do hiperespaço são algoritmos das bibliotecas `PyIT-MLFS` (ESKANDARI, 2022) e `scikit-learn` (PEDREGOSA et al., 2011), e algoritmos desenvolvidos neste trabalho, conforme descrito no Capítulo 7. Os al-

goritmos de classificação multirrótulo são os mesmos algoritmos da estratégia EMANUEL, são implementações disponíveis nas bibliotecas `WEKA` (HALL et al., 2009) e `MEKA` (READ et al., 2016). Além disso, EMANUEL$_{FS}$ inclui os algoritmos MLkNN, HOMER e ECC a seu hiperespaço de classificação multirrótulo. Esses algoritmos foram incluídos no *wrapper* da biblioteca `MEKA` para a biblioteca `MULAN` (TSOUMAKAS et al., 2011).

A implementação de EMANUEL$_{FS}$ foi feita nos moldes de EMANUEL. EMANUEL$_{FS}$ utiliza o NSGA-II como algoritmo de otimização, com os hiperparâmetros descritos na metodologia experimental (Seção 4.4). O objetivo da otimização é minimizar simultaneamente $-1 *$ Macro F-score e o tamanho do modelo, ou seja, encontrar modelos menores e mais simples, mas com bom desempenho em relação ao Macro F-score. A população inicial é gerada de forma aleatória e a decodificação de um indivíduo em algoritmo de seleção de atributos e de classificação multirrótulo segue a lógica descrita na Seção 8.3.

De forma similar à EMANUEL, EMANUEL$_{FS}$ também limita o tempo de execução dos algoritmos de seleção de atributos e de classificadores multirrótulo (esses limites são aplicados aos três folds). Limitar o tempo de execução de algoritmos no AutoML é uma prática comum. Quando os algoritmos excedem os tempos pré-estabelecidos, os objetivos da otimização multiobjetivo recebem os valores subótimos: zero para $-1 *$ Macro F-score e 1GB para o tamanho do modelo. Além disso, a estratégia EMANUEL$_{FS}$ também armazena os valores dos objetivos das soluções candidatas (algoritmo de seleção de atributos e algoritmo de classificação multirrótulo) já avaliadas. Assim, antes de avaliar uma nova solução, EMANUEL$_{FS}$ consulta o histórico de execuções para evitar que uma mesma solução seja avaliada mais de uma vez.

8.5 *Baseline* e *Setup* de Execução

Para os experimentos, foram selecionados onze conjuntos de dados do repositório `MULAN`: *bibtex*; *birds*; *cal500*; *corel5k*; *emotions*; *enron*; *flags*; *genbase*; *medical*; *scene*; e *yeast*. Todos os conjuntos de dados passaram pelo pré-processamento manual seguindo a metodologia descrita na Seção 4.2. Adotamos o pré-processamento com base nos resultados do experimento sobre seleção de atributos em dados multirrótulo, descritos no Capítulo 7. Os resultados demonstraram que o algoritmo de aprendizado alcançou desempenho superior quando aplicados a conjuntos de dados pré-processados, em comparação com os dados em seu estado original.

Executamos EMANUEL$_{FS}$ limitando o tempo de execução do algoritmo de seleção de atributos a cinco minutos e do algoritmo de classificação multirrótulo a 10 minutos. No experimento descrito no Capítulo 5, sobre a estratégia EMANUEL, o tempo limite adotado para os classificadores multirrótulo foi de 30 minutos. No entanto, diminuímos esse tempo para 10 minutos, com o intuito de diminuir a complexidade de tempo da estratégia proposta uma vez que EMANUEL$_{FS}$ computa adicionalmente a seleção de

atributos antes da indução dos modelos de classificação.

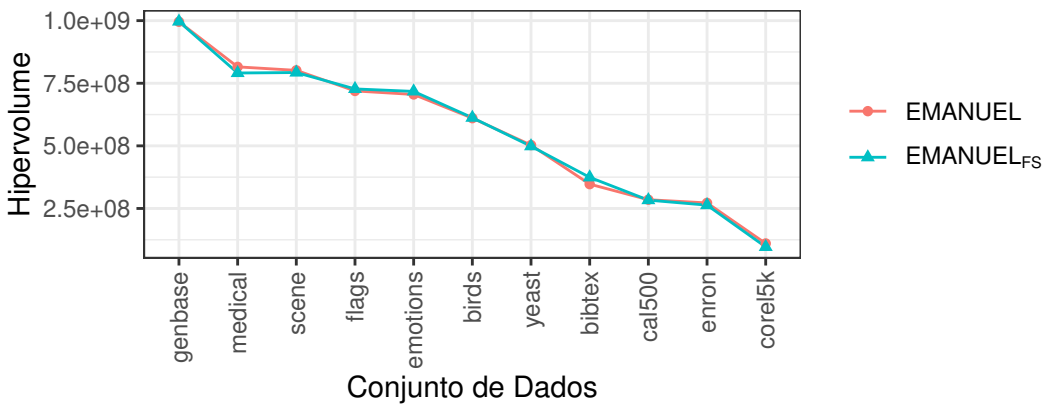
Comparamos EMANUEL$_{FS}$ com EMANUEL, pois EMANUEL foi a estratégia com os melhores resultados dentre os *baselines* comparados nos experimentos descritos nos Capítulos 5 e 6. Além disso, EMANUEL$_{FS}$ é uma extensão de EMANUEL, o que o torna um bom *baseline* de comparação. Para garantir uma comparação justa, utilizamos o mesmo hiperespaço de classificação multirrótulo em ambas as estratégias. Para isso, incluímos os algoritmos de classificação MLkNN, HOMER e ECC ao hiperespaço de EMANUEL. Executamos EMANUEL nos conjuntos de dados pré-processados supracitados, limitando o tempo de execução dos algoritmos de classificação multirrótulo a 10 minutos. Executamos ambas as estratégias cinco vezes para cada conjunto de dados.

8.6 Resultados

Utilizando as soluções dispostas nas cinco fronteiras de Pareto, resultantes da execução da estratégia EMANUEL$_{FS}$ para um conjunto de dados, encontramos a primeira fronteira de Pareto não dominada. Essa fronteira agrega as soluções de diferentes execuções, filtrando aquelas que mais otimizam os objetivos. Assim, das cinco fronteiras iniciais, temos uma fronteira resultante. Este procedimento também foi realizado para a estratégia EMANUEL. As fronteiras resultantes de ambas as estratégias foram utilizadas para a discussão dos resultados.

O hipervolume produzido pelas fronteiras resultantes é um indicativo de qualidade dos resultados encontrados. Valores mais altos do hipervolume indicam que o algoritmo de otimização encontrou soluções que otimizam mais os valores dos objetivos. A Figura 54 apresenta os hipervolumes das fronteiras resultantes de EMANUEL e de EMANUEL$_{FS}$. Na figura, os hipervolumes se sobrepõem para a maioria dos conjuntos de dados, sendo um indicativo de que o comportamento das estratégias é parecido. Ou seja, há um indicativo de que EMANUEL e EMANUEL$_{FS}$ encontram soluções parecidas.

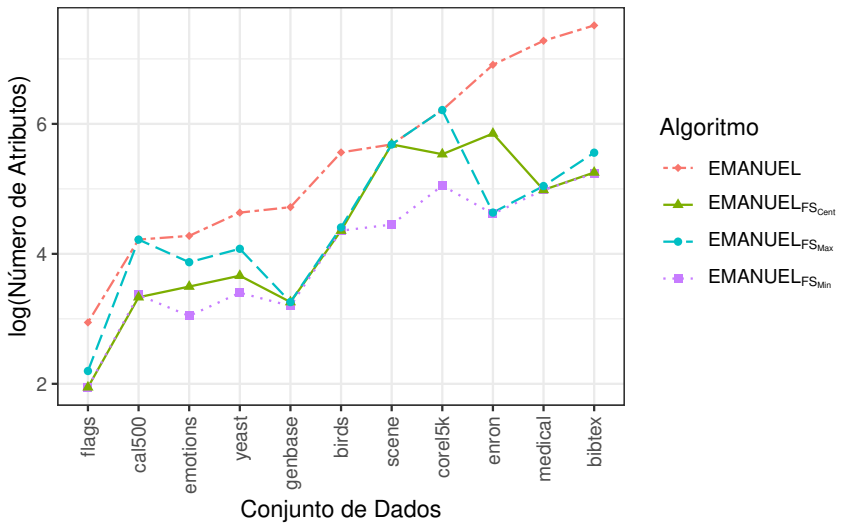
Figura 54 – Hipervolume resultante das estratégias EMANUEL e EMANUEL$_{FS}$.



Para avaliar as soluções das fronteiras resultantes de EMANUEL e de EMANUEL$_{FS}$, selecionamos as soluções das fronteiras com valores mínimos (Min), medianos ou centrais (Cent) e máximos (Max) dos objetivos. As soluções de EMANUEL foram nomeadas: EMANUEL$_{\text{Min}}$, EMANUEL$_{\text{Cent}}$ e EMANUEL$_{\text{Max}}$. Já as soluções de EMANUEL$_{FS}$ foram nomeadas: EMANUEL$_{\text{FS}_{\text{Min}}}$, EMANUEL$_{\text{FS}_{\text{Cent}}}$ e EMANUEL$_{\text{FS}_{\text{Max}}}$. Esperávamos que as soluções de EMANUEL$_{FS}$ mantivessem ou até melhorassem o desempenho de EMANUEL em relação aos objetivos.

As soluções[1] selecionadas de EMANUEL utilizaram todos os atributos disponíveis nos conjuntos de dados. As soluções de EMANUEL$_{FS}$ podem ou não fazer a seleção de atributos. A Figura 55 mostra o número de atributos utilizados pelas soluções selecionadas das fronteiras resultantes de EMANUEL e de EMANUEL$_{FS}$. Na figura, o número máximo de atributos é definido pela curva que representa EMANUEL. EMANUELFS$_{\text{FS}_{\text{Max}}}$ trabalhou com todos os atributos dos conjuntos de dados *cal500*, *corel5k* e *scene*. Já EMANUELFS$_{\text{FS}_{\text{Cent}}}$ trabalhou com todos os atributos do conjunto de dados *scene*. Para as demais soluções selecionadas das fronteiras de EMANUEL$_{FS}$, o número de atributos foi reduzido em relação ao número total de atributos do conjunto de dados.

Figura 55 – Número de atributos utilizados pelos algoritmos selecionados das fronteiras resultantes de EMANUEL e de EMANUEL$_{FS}$. Os algoritmos selecionados da fronteira de EMANUEL utilizam todos os atributos do conjunto de dados.



Para ajudar a visualizar a diferença entre os números de atributos utilizados por cada algoritmo, a Tabela 13 mostra a quantidade média de atributos (média dos *folds*) correspondente aos valores representados na Figura 55. Os algoritmos de EMANUEL$_{FS}$ que não adotaram a seleção de atributos foram destacados por um sublinhado na tabela (coluna *Atributos* de *EMANUEL$_{FS}$)*. Na grande maioria dos casos, EMANUEL$_{FS}$ trabalha

[1] Para EMANUEL uma solução é um algoritmo de classificação multirrótulo e para EMANUEL$_{FS}$ ela é composta pelos algoritmos multirrótulo de seleção de atributos e de classificação.

com conjuntos de dados de dimensionalidade reduzida, diminuindo significativamente o número de atributos. Como ficará claro nas análises posteriores, conjuntos de dados com dimensionalidade reduzida nem sempre contribuem para melhorias nos valores de Macro F-score e do tamanho do modelo.

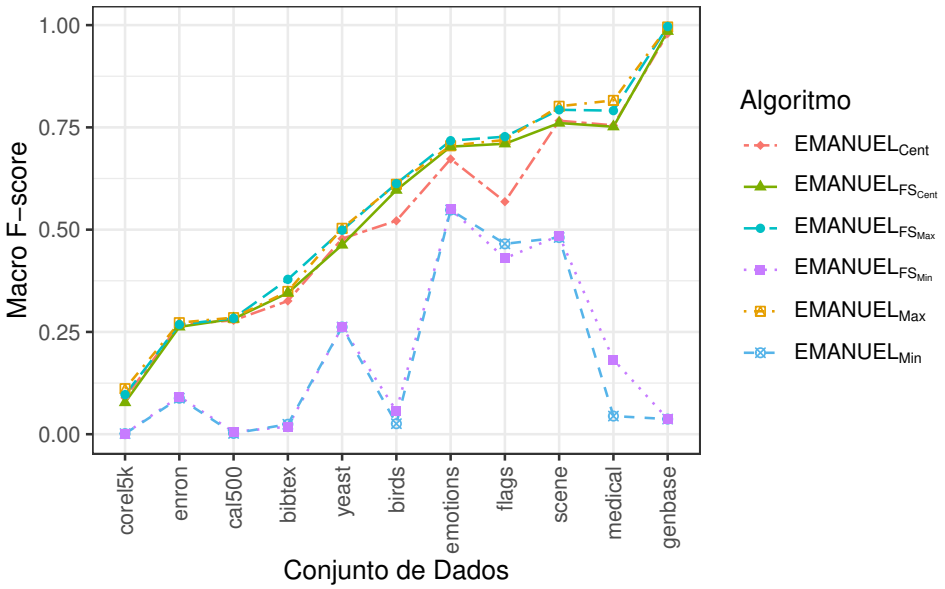
Tabela 13 – Resultados de EMANUEL e de EMANUEL$_{FS}$ para o número de atributos, o Macro F-score e o tamanho dos modelos. Para cada conjunto de dados, os melhores valores para o Macro F-score e o tamanho dos modelos estão destacados com um sublinhado. Os valores destacados para o número de atributos da estratégia EMANUEL$_{FS}$ referem-se aos casos onde não houve seleção de atributos.

Conj. de Dados	**Tipo**	**EMANUEL**			**EMANUEL$_{FS}$**		
		Atributos	**Mac. F-score**	**T. Modelo**	**Atributos**	**Mac. F-score**	**T. Modelo**
bibtex	Min		0.0243	3.74e+05	188	0.0183	6.71e+04
bibtex	Cent	1836	0.3257	1.44e+07	191	0.3453	3.91e+06
bibtex	Max		0.3490	6.48e+07	259	0.3785	2.09e+08
birds	Min		0.0257	2.95e+04	78	0.0570	1.24e+04
birds	Cent	260	0.5217	4.70e+05	78	0.5967	5.15e+05
birds	Max		0.6110	8.68e+06	82	0.6127	4.75e+07
cal500	Min		0.0020	3.96e+04	29	0.0057	3.37e+04
cal500	Cent	68	0.2783	4.58e+05	28	0.2820	2.54e+05
cal500	Max		0.2850	6.17e+06	68	0.2833	7.35e+07
corel5k	Min		0.0013	1.26e+05	156	0.0013	6.23e+04
corel5k	Cent	499	0.0890	8.22e+05	253	0.0775	9.00e+05
corel5k	Max		0.1113	1.03e+08	499	0.0967	1.42e+07
emotions	Min		0.5480	1.13e+04	21	0.5503	5.94e+03
emotions	Cent	72	0.6727	3.57e+04	33	0.7030	1.91e+05
emotions	Max		0.7053	2.67e+06	48	0.7175	9.73e+06
enron	Min		0.0880	1.91e+05	101	0.0920	2.88e+04
enron	Cent	1001	0.2680	5.45e+06	347	0.2623	1.89e+06
enron	Max		0.2727	6.09e+07	103	0.2680	2.20e+09
flags	Min		0.4653	6.15e+03	7	0.4300	4.32e+03
flags	Cent	19	0.5683	7.82e+03	7	0.7100	1.05e+04
flags	Max		0.7193	1.62e+04	9	0.7273	5.46e+04
genbase	Min		0.0370	2.45e+04	24	0.0375	9.16e+03
genbase	Cent	112	0.9793	4.06e+04	26	0.9850	1.75e+04
genbase	Max		0.9955	7.41e+05	26	0.9965	1.20e+05
medical	Min		0.0443	2.71e+05	146	0.1823	3.32e+04
medical	Cent	1449	0.7550	3.54e+05	146	0.7517	6.31e+04
medical	Max		0.8160	1.33e+07	155	0.7910	1.29e+05
scene	Min		0.4807	3.09e+04	85	0.4853	1.24e+04
scene	Cent	294	0.7663	9.07e+04	294	0.7607	9.61e+04
scene	Max		0.8017	2.40e+07	294	0.7930	1.44e+07
yeast	Min		0.2620	1.40e+04	30	0.2620	7.76e+03
yeast	Cent	103	0.4780	2.71e+05	39	0.4627	1.08e+05
yeast	Max		0.5033	1.11e+07	59	0.4987	6.40e+06

A Figura 56 mostra os valores de Macro F-score para as soluções selecionadas das fronteiras das estratégias avaliadas. No gráfico, as curvas demonstram similaridade entre os Macro F-scores de EMANUEL$_{\text{FS}_{\text{Min}}}$ e EMANUEL$_{\text{Min}}$, EMANUEL$_{\text{FS}_{\text{Cent}}}$

e EMANUEL$_{\text{Cent}}$, e entre EMANUEL$_{\text{FS}_{\text{Max}}}$ e EMANUEL$_{\text{Max}}$. Além disso, as soluções selecionadas no centro das fronteiras têm Macro F-score elevados, próximos às soluções EMANUEL$_{\text{FS}_{\text{Max}}}$ e EMANUEL$_{\text{Max}}$. Considerando a redução significativa no número de atributos, EMANUEL$_{FS}$ consegue manter Macro F-score similar ao encontrado por EMANUEL, nas diferentes soluções selecionadas (Min, Cent e Max) da fronteira de Pareto.

Figura 56 – Macro F-score das soluções selecionadas das fronteiras de EMANUEL e de EMANUEL$_{FS}$.

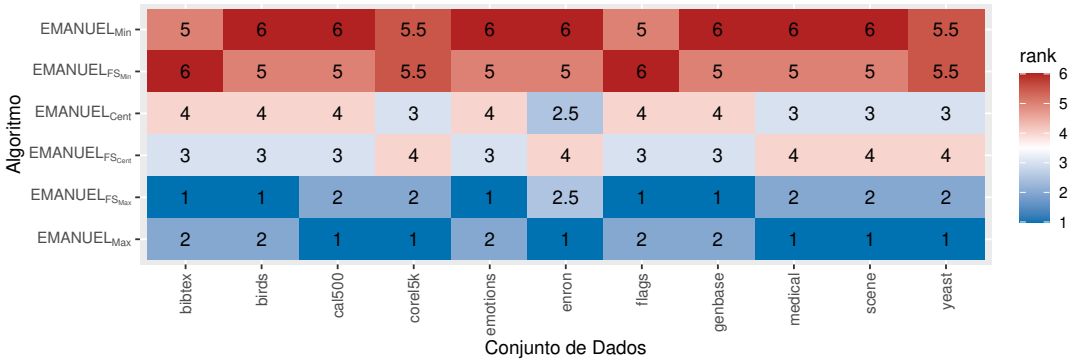


Para ajudar a visualizar os resultados de Macro F-score, na Figura 57 os algoritmos foram classificados em relação ao Macro F-score. Na figura, os menores *rankings* (quadrados azuis) representam os algoritmos com os melhores Macro F-scores e os algoritmos com os piores Macro F-scores têm os maiores *rankings* (quadrados vermelhos). Na figura, os algoritmos são apresentados pelo *rankings* médio.

Em média, EMANUEL$_{\text{Max}}$ e EMANUEL$_{\text{FS}_{\text{Max}}}$ são os algoritmos com maiores Macro F-scores (Figura 57). EMANUEL$_{\text{FS}_{\text{Max}}}$ tem desempenho superior a EMANUEL$_{\text{Max}}$ em seis conjuntos de dados. Enquanto EMANUEL$_{\text{Max}}$ supera EMANUEL$_{\text{FS}_{\text{Max}}}$ em cinco conjuntos de dados. EMANUEL$_{\text{FS}_{\text{Cent}}}$ e EMANUEL$_{\text{Cent}}$ são os próximos algoritmos do *ranking* médio. EMANUEL$_{\text{FS}_{\text{Cent}}}$ tem desempenho superior a EMANUEL$_{\text{cent}}$ em seis conjuntos de dados e EMANUEL$_{\text{cent}}$ é superior a EMANUEL$_{\text{FS}_{\text{Cent}}}$ em cinco conjuntos de dados. Entretanto, a diferença de desempenho entre esses pares de algoritmos é muito pequena, conforme dados da Tabela 13, o que pode indicar que EMANUEL e de EMANUEL$_{FS}$ não apresentam diferenças estatísticas significativas em relação ao Macro F-score.

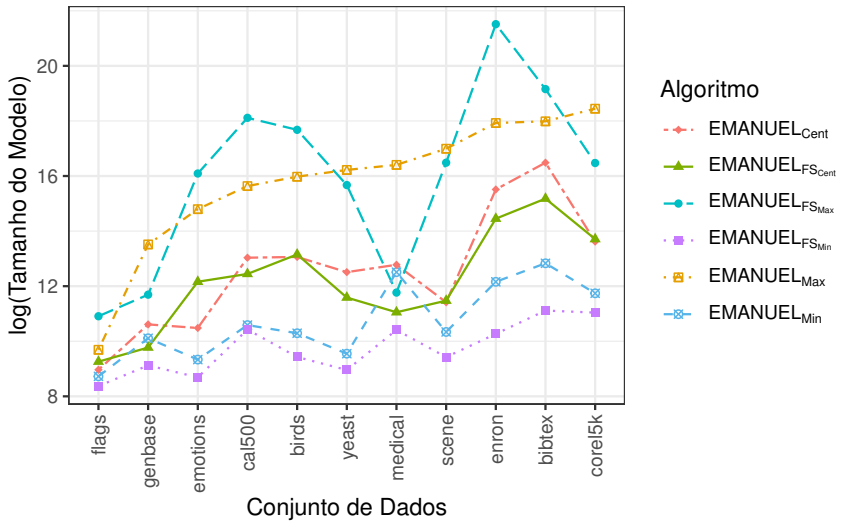
Considerando o tamanho dos modelos, a Figura 58 apresenta os tamanhos dos modelos induzidos pelos algoritmos selecionados das fronteiras de EMANUEL e de EMANUEL$_{FS}$.

Figura 57 – *Ranking* dos algoritmos selecionados das fronteiras de EMANUEL e de EMANUEL$_{FS}$ quanto ao Macro F-score. Na figura, os melhores *rankings* são os menores e *rankings* iguais para algoritmos indicam Macro F-score iguais.



Tanto EMANUEL$_{\text{FS}_{\text{Min}}}$ quanto EMANUEL$_{\text{Min}}$ são os algoritmos que induzem os menores modelos. Por outro lado, EMANUEL$_{\text{FS}_{\text{Cent}}}$ e EMANUEL$_{\text{Cent}}$ induzem modelos com tamanhos intermediários. Nota-se que EMANUEL$_{\text{FS}_{\text{Cent}}}$ induz modelos menores do que EMANUEL$_{\text{Cent}}$ em seis conjuntos de dados, enquanto EMANUEL$_{\text{Cent}}$ induz modelos menores do que EMANUEL$_{\text{FS}_{\text{Cent}}}$ em cinco conjuntos de dados. EMANUEL$_{\text{FS}_{\text{Max}}}$ e EMANUEL$_{\text{Max}}$ são os algoritmos que induzem os maiores modelos. EMANUEL$_{\text{Max}}$ induz modelos maiores do que EMANUEL$_{\text{FS}_{\text{Max}}}$ em cinco conjuntos de dados, enquanto EMANUEL$_{\text{FS}_{\text{Max}}}$ supera EMANUEL$_{\text{Max}}$ em seis conjuntos de dados. Para mais informações sobre o tamanho dos modelos, os valores numéricos ilustrados na Figura 58 encontram-se na Tabela 13.

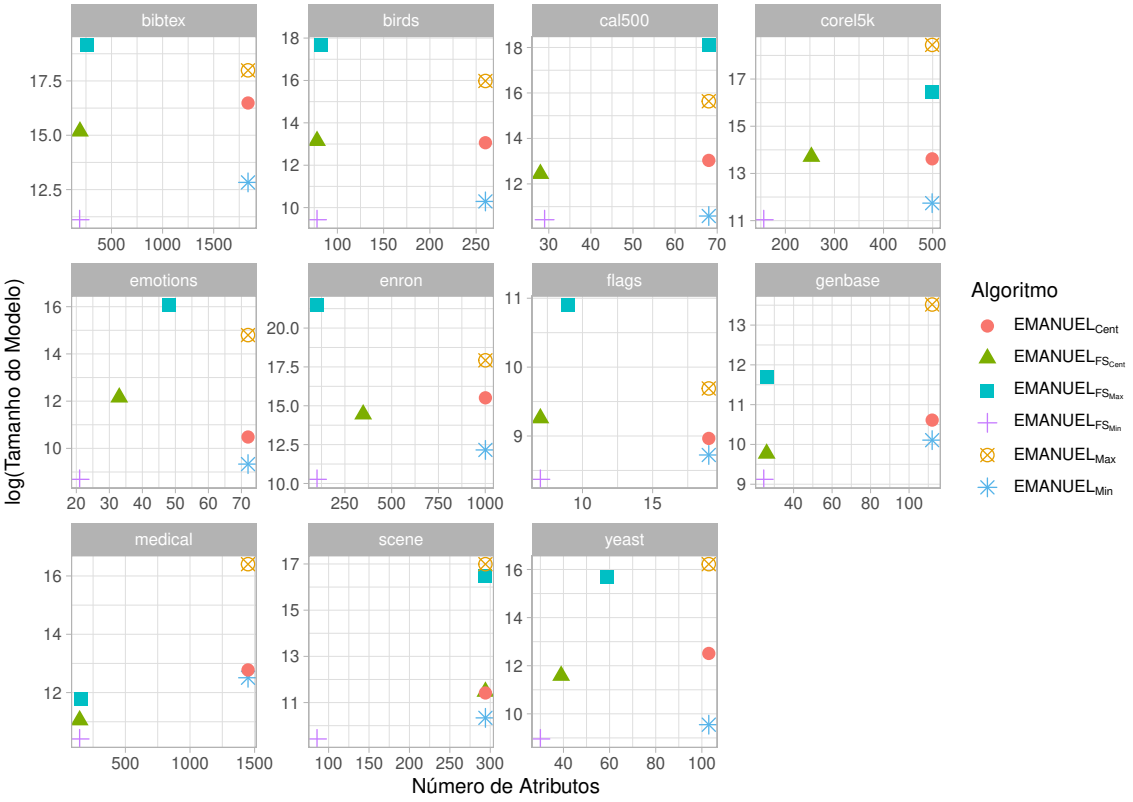
Figura 58 – Tamanhos dos modelos induzidos pelos algoritmos selecionado das fronteiras de EMANUEL e de EMANUEL$_{FS}$.



Na Figura 58, as curvas demonstram similaridade entre os tamanhos dos modelos de EMANUEL$_{\text{FS}_{\text{Min}}}$ e de EMANUEL$_{\text{Min}}$ e entre EMANUEL$_{\text{FS}_{\text{Cent}}}$ e EMANUEL$_{\text{Cent}}$. Entretanto, essa similaridade não é observada entre EMANUEL$_{\text{FS}_{\text{Max}}}$ e EMANUEL$_{\text{Max}}$. Para os conjuntos de dados *cal500*, *medical* e *enron*, por exemplo, os tamanhos dos modelos resultantes de EMANUEL$_{\text{Max}}$ e EMANUEL$_{\text{FS}_{\text{Max}}}$ são bem diferentes.

EMANUEL$_{\text{FS}_{\text{Max}}}$ pode diminuir a dimensionalidade dos dados e encontrar modelos de tamanhos menores e com bons desempenhos, mas também pode encontrar modelos maiores. Essa variação pode estar relacionada a dois fatores. O primeiro é que EMANUEL$_{\text{FS}_{\text{Max}}}$ trabalha com a opção de não realizar a seleção de atributos, permitindo a indução de modelos que utilizam todos os atributos do conjunto de dados. Nesse caso, EMANUEL$_{FS}$ tem o mesmo comportamento de EMANUEL. O segundo fator está relacionado à seleção de atributos, que pode permitir que algoritmos de classificação mais complexos ou com configurações de hiperparâmetros mais elaboradas participem do AutoML, já que, com um conjunto de dados com dimensionalidade reduzida, esses algoritmos podem produzir resultados dentro do tempo limite estipulado para execução.

Figura 59 – Relacionamento entre o tamanho do modelo e o número de atributos resultantes das estratégias EMANUEL e EMANUEL$_{FS}$.



Para compreender melhor o comportamento de EMANUEL$_{FS}$ em relação ao número de atributos e ao tamanho do modelo, a Figura 59 exibe a relação entre essas duas variáveis. Os gráficos da figura podem ser subdivididos em quatro quadrantes. O primeiro e segundo

quadrantes estão na parte superior do gráfico, à direita e à esquerda, respectivamente. O terceiro e quarto quadrantes estão na parte inferior do gráfico, à esquerda e à direita, respectivamente.

$\text{EMANUEL}_{\text{FS}_{\text{Min}}}$ está no terceiro quadrante para todos os conjuntos de dados, representando um algoritmo que induz modelos pequenos, utilizando um número de atributos reduzido. Entretanto, esse algoritmo sempre apresenta valores ruins para o Macro F-score. $\text{EMANUEL}_{\text{FS}_{\text{Cent}}}$ também está no terceiro quadrante na maioria dos conjuntos. Ele induz modelos maiores e com Macro F-scores melhores do que $\text{EMANUEL}_{\text{FS}_{\text{Min}}}$. Esses algoritmos conseguem diminuir a dimensionalidade dos dados e encontrar modelos de tamanhos menores e com bom desempenho. $\text{EMANUEL}_{\text{FS}_{\text{Cent}}}$ encontra-se no quarto quadrante apenas para o conjunto de dados *scene*, pois para esse conjunto de dados $\text{EMANUEL}_{\text{FS}_{\text{Cent}}}$ não realiza seleção de atributos.

$\text{EMANUEL}_{\text{FS}_{\text{Max}}}$ encontra-se no terceiro quadrante apenas para o conjunto de dados *medical*. Para os conjuntos de dados *cal500*, *corel5k* e *scene*, $\text{EMANUEL}_{\text{FS}_{\text{Max}}}$ encontra-se no primeiro quadrante. Nesse quadrante estão os algoritmos que utilizam uma quantidade maior ou todos os atributos do conjunto de dados e induzem os maiores modelos. $\text{EMANUEL}_{\text{FS}_{\text{Max}}}$ encontra-se no segundo quadrante para os conjuntos de dados *bibtex*, *birds*, *enron*, *flags*, *genbase* e *yeast*. Nesse quadrante estão os algoritmos que utilizam menos atributos para a indução dos modelos, mas os modelos induzidos são grandes, provavelmente devido ao algoritmo de classificação multirrótulo empregado. $\text{EMANUEL}_{\text{FS}_{\text{Max}}}$ encontra-se entre o primeiro e o segundo quadrantes para o conjunto de dados *emotions*. Nesse caso, $\text{EMANUEL}_{\text{FS}_{\text{Max}}}$ utiliza uma quantidade mediana de atributos, mas também induz um modelo de tamanho grande.

EMANUEL sempre utiliza todos os atributos. Assim, na Figura 59, seus resultados para os diferentes conjuntos de dados sempre estão no primeiro e quarto quadrantes, pois esses quadrantes representam soluções com elevado número de atributos. $\text{EMANUEL}_{\text{Max}}$ sempre induz modelos grandes e, para todos os conjuntos de dados, está no primeiro quadrante. $\text{EMANUEL}_{\text{Min}}$ sempre induz modelos menores e, para todos os conjuntos de dados, está no quarto quadrante. Os modelos induzidos por $\text{EMANUEL}_{\text{Cent}}$ alternam entre o primeiro e o quarto quadrantes, estando no quarto quadrante na maioria dos conjuntos de dados.

De acordo com os resultados discutidos até o momento, os tamanhos dos modelos podem ser discrepantes entre os pares de soluções selecionadas (*Min* e *Min*, *Cent* e *Cent*, *Max* e *Max*) de $\text{EMANUEL}_{\text{Max}}$ e de $\text{EMANUEL}_{\text{FS}_{\text{Min}}}$. Além disso, ambas as estratégias alternam entre os melhores e piores resultados para o Macro F-score e para o tamanho do modelo, dependendo da solução selecionada. Portanto, aplicamos o teste de Wilcoxon para verificar se as estratégias propostas apresentavam diferenças estatisticamente significativas em termos do Macro F-score e do tamanho do modelo. A hipótese nula não foi rejeitada no teste de Wilcoxon para o Macro F-score ($p\text{-}value = 0.416$) e no teste de Wilcoxon para

o tamanho do modelo (p-*value* = 0.195), indicando que não há diferenças significativas entre os resultados de EMANUEL e de EMANUEL$_{FS}$.

Para verificar se os algoritmos selecionados das fronteiras de Pareto apresentavam desempenhos estatisticamente significativos em termos dos objetivos da otimização, aplicamos o teste de Friedman. A hipótese nula foi rejeitada para ambos os testes estatísticos, indicando diferenças significativas entre os algoritmos analisados. Para identificar os grupos de algoritmos com desempenhos distintos, aplicamos o teste *post-hoc* de Nemenyi. Os resultados para o Macro F-score são ilustrados no diagrama de diferença crítica da Figura 60. Nesse diagrama, algoritmos interligados por uma linha não apresentam diferenças estatisticamente significativas entre si. No diagrama, os algoritmos *Max* e *Cent* selecionados das fronteiras de EMANUEL e de EMANUEL$_{FS}$ apresentam os melhores Macro F-scores. A Figura 61 ilustra o diagrama de diferença crítica para o tamanho do modelo. No diagrama, EMANUEL$_{\text{FS}_{\text{Min}}}$, EMANUEL$_{\text{Min}}$ e EMANUEL$_{\text{FS}_{\text{Cent}}}$ são os algoritmos com os melhores tamanhos de modelos.

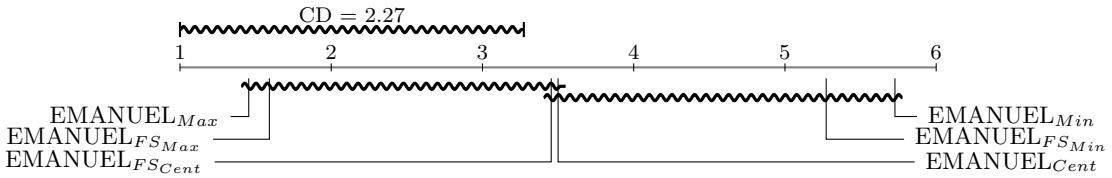


Figura 60 – Diagrama de diferença crítica para o teste de Nemenyi para o Macro F-score.

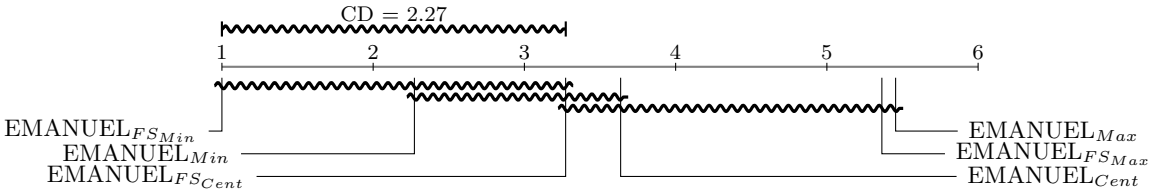


Figura 61 – Diagrama de diferença crítica para o teste de Nemenyi para o tamanho do modelo.

Do ponto de vista multiobjetivo, as soluções selecionadas no centro das fronteiras de Pareto (EMANUEL$_{\text{Cent}}$ e EMANUEL$_{\text{FS}_{\text{Cent}}}$) são as melhores, pois elas alcançam bons resultados tanto para o Macro F-score quanto para o tamanho do modelo, conforme ilustram as Figuras 60 e 61. Na Figura 60, EMANUEL$_{\text{Cent}}$ e EMANUEL$_{\text{FS}_{\text{Cent}}}$ estão no grupo de algoritmos com os melhores Macro F-scores. Já na Figura 61, EMANUEL$_{\text{FS}_{\text{Cent}}}$ pertence ao grupo de algoritmos com menores tamanhos de modelos e EMANUEL$_{\text{Cent}}$ não pertence a este grupo. No entanto, o gráfico da Figura 61 mostra uma pequena diferença de classificação entre EMANUEL$_{\text{Cent}}$ e EMANUEL$_{\text{FS}_{\text{Cent}}}$, sendo inclusive estatisticamente equivalentes entre si.

8.7 Considerações Finais

Neste capítulo propomos EMANUEL$_{FS}$, uma extensão de EMANUEL que inclui a função de seleção de atributos à estratégia AutoML. EMANUEL$_{FS}$ busca a combinação de algoritmos de seleção de atributos e de classificação multirrótulo que maximizam o Macro F-score e minimizam o tamanho do modelo simultaneamente. Para verificar o desempenho de EMANUEL$_{FS}$, o comparamos com EMANUEL. Estatisticamente, o Macro F-score e o tamanho dos modelos induzidos por EMANUEL$_{FS}$ e EMANUEL não apresentam diferenças estatísticas significativas. Assim, nossa hipótese de pesquisa foi confirmada pelos resultados experimentais.

Até onde sabemos, EMANUEL$_{FS}$ é a primeira estratégia AutoML destinada especificamente para problemas multirrótulo, que inclui seleção de atributos. Entretanto, a complexidade da estratégia EMANUEL$_{FS}$ é maior do que a complexidade de EMANUEL, pois ela inclui a seleção de atributos como etapa adicional no processo AutoML, realizada antes da indução dos classificadores. Assim como ocorre em EMANUEL$_{SM}$, os modelos substitutos poderiam ser utilizados na estratégia EMANUEL$_{FS}$ para prever o desempenho de cada solução candidata (composta pelos algoritmos de seleção de atributos e de classificação), o que reduziria o tempo de execução. Dessa forma, o próximo capítulo apresenta uma prova de conceito sobre uso de modelos substitutos em EMANUEL$_{FS}$.

Capítulo 9

EMANUEL$_{\mathbf{FS+}}$

No Capítulo 8, apresentamos EMANUEL$_{FS}$, uma estratégia AutoML que encontra automaticamente soluções para problemas de classificação multirrótulo que maximizam o desempenho (Macro F-score) e que minimizam a complexidade dos modelos (representada pelo tamanho dos modelos). Nessa estratégia, cada solução é composta por um algoritmo de seleção de atributos, um algoritmo de classificação multirrótulo e a correspondente configuração de hiperparâmetros desses algoritmos.

O mecanismo de busca de EMANUEL$_{FS}$ é baseado em um algoritmo evolutivo multi-objetivo que explora o espaço de soluções. Durante o processo evolutivo, a estratégia avalia iterativamente as soluções candidatas, executando tanto a seleção de atributos quanto a indução do modelo de classificação. Essa busca progressiva permite a construção de uma fronteira de Pareto, contendo as soluções que mais ponderam os objetivos da otimização.

Entretanto, a avaliação de uma solução candidata pode ser computacionalmente cara e, no contexto do AutoML, essa complexidade é ainda maior. Conforme já mencionado no Capítulo 6, uma alternativa para essa questão é o uso do meta-aprendizado, mais especificamente de modelos substitutos, pois esses modelos são representações simplificadas de modelos mais complexos e são utilizados quando não há uma modelagem exata viável ou quando os modelos originais são computacionalmente caros.

Desta forma, propomos a estratégia denominada *gEnetic Multi-objective strategy for the Automatic selectioN of mUlti-labEl cLassifiers and Feature Selection algorithms based on surrogate models* (EMANUEL$_{FS+}$), que é uma extensão da estratégia EMANUEL$_{FS}$ e utiliza modelos substitutos para prever os valores dos objetivos da otimização. Nossa hipótese de pesquisa pressupõe que os modelos substitutos fornecem uma boa aproximação dos valores do Macro F-score e do tamanho do modelo ao avaliar uma solução candidata do AutoML, permitindo que a estratégia EMANUEL$_{FS+}$ mantenha o desempenho de

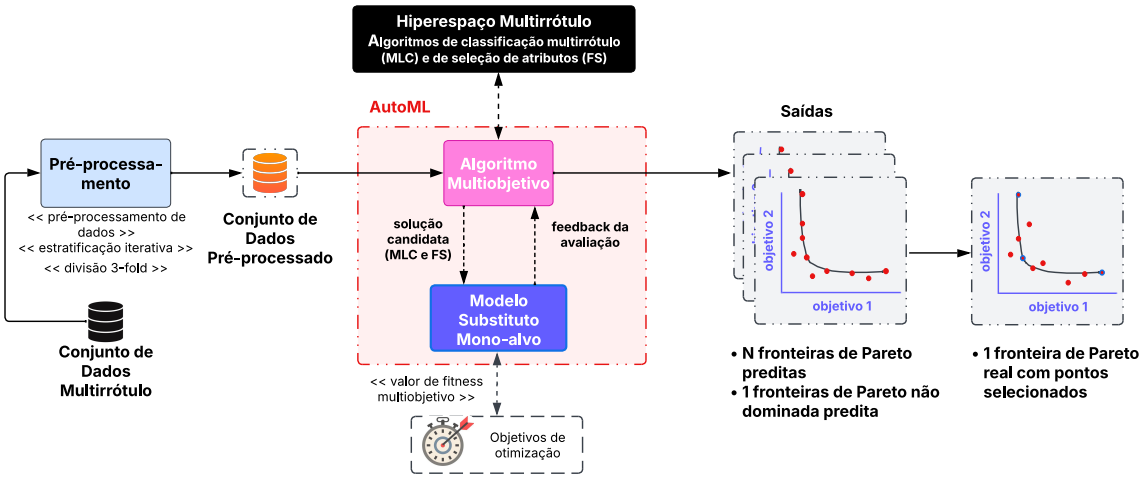
EMANUEL$_{FS}$ e reduza significativamente o tempo de execução. Para validar essa hipótese, conduzimos uma prova de conceito, com experimentos simplificados, utilizando um número reduzido de conjuntos de dados.

As próximas seções deste capítulo estão organizadas da seguinte forma: a Seção 9.1 descreve o funcionamento de EMANUEL$_{FS+}$; a Seção 9.2 apresenta os metadados; a Seção 9.3 descreve os modelos substitutos; a Seção 9.4 detalha a implementação da estratégia proposta; a Seção 9.5 identifica os *baselines* de comparação e o *setup* de execução; a Seção 9.6 apresenta os resultados; e a Seção 9.7 traz as considerações finais.

9.1 Como EMANUEL$_{FS+}$ Trabalha?

O funcionamento de EMANUEL$_{FS+}$, ilustrado na Figura 62, é similar ao da estratégia EMANUEL$_{FS}$. Porém, com diferença na etapa de avaliação. A estratégia recebe como entrada um conjunto de dados pré-processado manualmente. O algoritmo de otimização multiobjetivo seleciona do hiperespaço uma solução candidata. Neste experimento, o algoritmo de otimização foi o NSGA-II. NSGA-II seleciona do hiperespaço uma população de indivíduos. Cada indivíduo é decodificado em uma solução candidata que consiste em um algoritmo de seleção de atributos e um algoritmo de classificação multirrótulo.

Figura 62 – Fluxograma do funcionamento da estratégia EMANUEL$_{FS+}$.



Diferentemente do que ocorre com EMANUEL$_{FS}$, a solução candidata não é avaliada pelo avaliador, mas sim pelos modelos substitutos, que preveem os valores dos objetivos (Macro F-score e tamanho do modelo) e retornam um *feedback* da avaliação ao algoritmo de otimização. O processo de otimização continua até que se atinja uma condição de parada pré-definida. A vantagem da nova estratégia é que ela elimina a necessidade da seleção de atributos e do treinamento do classificador para o cálculo dos objetivos. Em

vez disso, os valores dos objetivos são estimados pelos modelos substitutos, o que torna a execução de EMANUEL$_{FS+}$ significativamente mais rápida.

Após N execuções da estratégia EMANUEL$_{FS+}$, há N fronteiras de Pareto distintas. Para incorporar as N fronteiras em uma única fronteira representativa, aplicamos o conceito de não dominância (DEB et al., 2002), identificando a primeira fronteira não dominada. No entanto, a primeira fronteira não dominada deriva-se dos valores previstos. Cada ponto da fronteira corresponde a uma combinação de algoritmos de seleção de atributos e de classificação. Executamos os algoritmos correspondentes de cada ponto da fronteira e reconstruímos a primeira fronteira não dominada a partir de valores reais. Por fim, selecionamos três algoritmos da fronteira de Pareto real como resultados de EMANUEL$_{FS+}$, utilizando *Frugality Score*, conforme descrito na Seção 5.1.4.

9.2 Criando os Metadados

Os metadados contêm o histórico das execuções dos classificadores multirrótulo quando induzidos em conjuntos de dados de menor dimensionalidade, resultantes da seleção de atributos. Assim como ocorre em EMANUEL$_{SM}$ (Seção 6.2), criamos dois meta-conjuntos de dados para cada conjunto de dados. Um foi gerado a partir da execução da estratégia EMANUEL$_{FS}$ (MtDt$_{\mathrm{A}}$) e o outro meta-conjunto de dados foi criado a partir da execução do algoritmo MORS (MtDt$_{\mathrm{R}}$). Ambos os meta-conjuntos de dados incluem todos os algoritmos de seleção e de classificação avaliados durante as respectivas estratégias de busca.

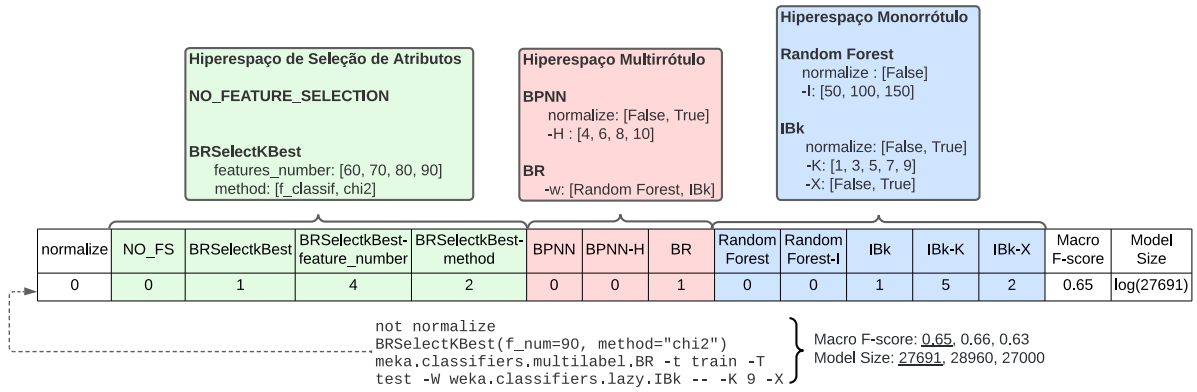
Tanto EMANUEL$_{FS}$ quanto MORS utilizaram os hiperespaços definidos nos Apêndices A e F para delimitar os possíveis algoritmos dos meta-conjuntos de dados. Além disso, ambas as estratégias de busca limitaram o tempo de execução dos algoritmos de seleção e de classificação a cinco e 10 minutos, respectivamente. Durante a geração dos metadados, os algoritmos de seleção de atributos e de classificação foram executados nos *3-folds* pré-definidos. Como resultados desses algoritmos, foram considerados os valores medianos dos *3-folds* tanto do Macro F-score quanto do tamanho do modelo.

Nos meta-conjuntos de dados, cada meta-atributo de entrada representa um algoritmo ou um hiperparâmetro dos hiperespaços de seleção de atributos e de classificação. O primeiro meta-atributo representa a normalização. Em seguida, os meta-atributos representam os algoritmos de seleção de atributos e seus respectivos hiperparâmetros. Por fim, os meta-atributos representam os algoritmos de classificação multirrótulo e monorrótulo, bem como os seus hiperparâmetros.

Os meta-atributos de entrada são dispostos no meta-conjunto de dados na mesma ordem em que são definidos nos hiperespaços de seleção de atributos e de classificação. Os valores dos meta-atributos que representam a normalização ou um algoritmo são binários. Já os valores dos meta-atributos que representam os hiperparâmetros são números inteiros

correspondentes aos índices de vetores de hiperparâmetros (iniciando em 1) definidos nos hiperespaços. Os meta-atributos alvo dos meta-conjuntos de dados são os valores medianos do Macro F-score e do logaritmo do tamanho do modelo.

Figura 63 – Exemplo de um meta-conjunto de dados para hiperespaços de seleção de atributos e de classificação reduzidos. Na figura, `NO_FS` representa a opção `NO_FEATURE_SELECTION`.



A Figura 63 exemplifica como os meta-conjuntos de dados são criados, considerando hiperespaços de seleção de atributos e de classificação reduzidos. A disposição dos meta-atributos de entrada é definida com base nos hiperespaços, conforme descrito acima. Assim, nesse exemplo, a disposição dos meta-atributos de entrada é: `normalize`, `NO_FEATURE_SELECTION`, `BRSelectkBest`, `BRSelectkBest-features_number`, `BRSelectkBest-method`, `BPNN`, `BPNN-H`, `BR`, `Random Forest`, `Random Forest-I`, `IBk`, `IBk-k` e `IBk-X`. Os meta-atributos alvos do meta-conjunto de dados são os valores medianos do Macro F-score e do logaritmo do tamanho do modelo resultantes da execução dos algoritmos de seleção e de classificação nos *3-folds* (0.65 e $log(27691)$, respectivamente). A figura também ilustra a codificação dos algoritmos de seleção e de classificação para uma instância do meta-conjunto de dados. Considerando o hiperespaço adotado por EMANUEL$_{FS+}$ (descrito na Seção 8.2), o meta-conjunto de dados criado possui 209 meta-atributos de entrada e dois meta-atributos alvos preditivos relacionados aos objetivos da otimização multiobjetivo.

9.3 Os Modelos Substitutos

O procedimento para criação dos modelos substitutos da estratégia EMANUEL$_{FS+}$ foi o mesmo adotado para os modelos substitutos de EMANUEL$_{SM}$. Os modelos substitutos foram induzidos com o algoritmo RF da biblioteca `scikit-learn` com hiperparâmetros padrão (`n_estimators = 100`) (PEDREGOSA et al., 2011). Esses modelos estimam os

valores de Macro F-score e do logaritmo dos tamanhos dos modelos, podendo ser do tipo mono-alvo ou multi-alvo e induzidos com os diferentes meta-conjuntos de dados. Conduzimos dois experimentos para definir a configuração ideal dos modelos substitutos. O primeiro experimento repetiu a indução de modelos RF mono-alvo e multi-alvo 10 vezes *10-folds* no meta-conjunto de dados resultante da união de MtDt_A e MtDt_R ($\text{MtDt}_\text{A+R}$). De acordo com o teste de Wilcoxon, não há diferenças estatísticas entre os modelos avaliados em termos de R^2. No entanto, os modelos mono-alvo apresentaram R^2 ligeiramente superiores aos modelos multi-alvo em termos de R^2. Portanto, treinamos dois modelos substitutos, um para o Macro F-score e outro para o logaritmo do tamanho do modelo.

O segundo experimento avaliou o melhor meta-conjunto de dados para a indução dos modelos substitutos (MtDt_A, MtDt_R ou $\text{MtDt}_\text{A+R}$). Modelos RF mono-alvo foram induzidos nos diferentes meta-conjuntos de dados, utilizando a abordagem de validação cruzada *10-fold* com 10 repetições. Realizamos o teste de Friedman, considerando como medida de avaliação o R^2 médio dos modelos mono-alvo. $\text{MtDt}_\text{A+R}$ obteve R^2 médio estatisticamente superior aos demais meta-conjuntos de dados avaliados. Portanto, $\text{MtDt}_\text{A+R}$ foi o meta-conjunto de dados utilizado para a indução dos modelos substitutos mono-alvo. Seguindo essas configurações, dois modelos substitutos foram treinados para cada conjunto de dados multirrótulo. A Seção C.2 do Apêndice C descreve os experimentos em relação aos modelos substitutos com mais detalhes.

9.4 Detalhes de Implementação

A estratégia EMANUEL_{FS+} foi desenvolvida conforme a estratégia EMANUEL_{FS} (Seção 8.4), mantendo os mesmos elementos de sua estrutura: o hiperespaço (Seção 8.2), a representação dos indivíduos (Seção 8.3), o algoritmo de otimização multiobjetivo NSGA-II (com a configuração de hiperparâmetros definida na Seção 4.4), o Macro F-score e o tamanho do modelo como objetivos da otimização. A principal diferença entre as estratégias está nos modelos substitutos. EMANUEL_{FS} faz a seleção de atributos, induz os modelos de classificação multirrótulo e calcula o valor médio dos objetivos. Já EMANUEL_{FS+} utiliza os modelos substitutos para prever o valor mediano dos objetivos. Todas as vezes que o NSGA-II recebe uma estimativa do tamanho do modelo, o valor do logaritmo é convertido para seu valor original. Portanto, na estratégia EMANUEL_{FS} a evolução do NSGA-II ocorre com os valores médios dos objetivos, enquanto na estratégia EMANUEL_{FS+} ela ocorre com os valores medianos previstos pelos modelos substitutos.

9.5 Algoritmos *Baselines* e *Setup* de Execução

Os experimentos utilizaram os conjuntos de dados *birds*, *enron*, *medical*, *scene* e *yeast* obtidos do repositório `MULAN`. Essa seleção limitada de conjuntos de dados justifica-se pelo

caráter de prova de conceito deste capítulo. Esses conjuntos de dados foram escolhidos, pois entre os listados na Seção 4.2, eles possuem quantidade mediana de atributos e garantem tempos de execução viáveis para os experimentos. Todos os conjuntos de dados foram pré-processados manualmente, seguindo a metodologia descrita na Seção 4.2. Para fins comparativos, avaliamos o desempenho de EMANUEL$_{FS+}$ em relação às estratégias AutoML: EMANUEL, EMANUEL$_{FS}$ e `auto-sklearn`. Optamos por não incluir a estratégia EMANUEL$_{SM}$ como um *baseline*, pois os modelos substitutos dessa estratégia foram treinados em conjuntos de dados em seu formato original, sem pré-processamento.

EMANUEL e EMANUEL$_{FS}$ são importantes *baselines* de comparação, pois são a base para a construção de EMANUEL$_{FS+}$. EMANUEL automatiza apenas a busca pelo modelo de classificação, enquanto EMANUEL$_{FS}$ amplia essa funcionalidade, incorporando a seleção de atributos antes da busca pelo modelo de classificação. Para garantir uma comparação justa, EMANUEL, EMANUEL$_{FS}$ e EMANUEL$_{FS+}$ utilizaram os mesmos hiperespaços de classificação multirrótulo e monorrótulo. Quanto ao hiperespaço de seleção de atributos, este também foi idêntico para as estratégias EMANUEL$_{FS}$ e EMANUEL$_{FS+}$. Para essas estratégias AutoML, limitamos o tempo de execução do algoritmo de seleção de atributos a cinco minutos e do algoritmo de classificação multirrótulo a 10 minutos.

A ferramenta `auto-sklearn` foi outro importante *baseline* de comparação, pois ela automatiza a engenharia de atributos e a busca pelo modelo de classificação. Para executar `auto-sklearn`, desabilitamos o módulo de pré-processamento de dados e utilizamos os conjuntos de dados pré-processados manualmente. O módulo de pré-processamento de atributos foi mantido ativo. Para a execução do `auto-sklearn` utilizamos os hiperparâmetros definidos na Seção 4.5.1 e as especificações do manual da ferramenta, limitando o tempo total de execução a um dia e o tempo de execução de um algoritmo de classificação multirrótulo a 30 minutos (FEURER et al., 2015). No entanto, conforme mencionado na Seção 4.5.1, não podemos comparar `auto-sklearn` em relação ao tamanho dos modelos quando a automatização inclui outros módulos além da busca pelo modelo de AM. Tanto EMANUEL$_{FS+}$ quanto os demais algoritmos *baselines* foram executados cinco vezes para cada conjunto de dados.

9.6 Resultados

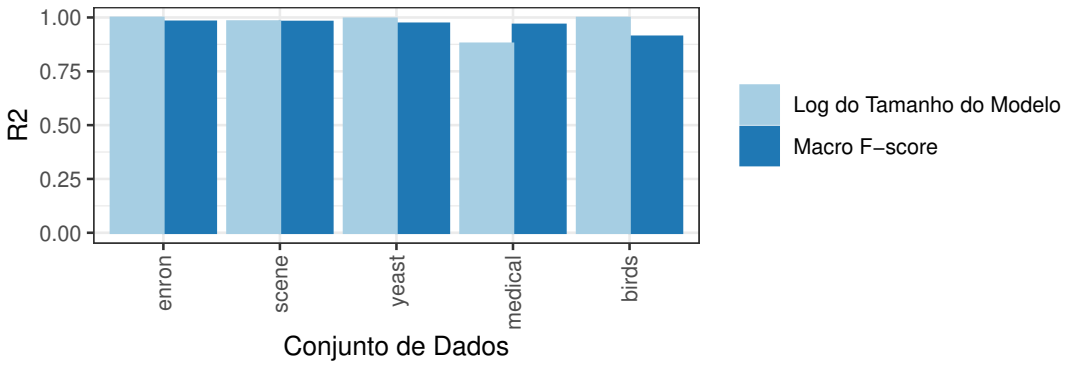
A partir das cinco execuções de EMANUEL, EMANUEL$_{FS}$ e EMANUEL$_{FS+}$ calculamos a primeira fronteira não dominada. Adicionalmente, para EMANUEL$_{FS+}$, recalculamos a primeira fronteira não dominada real. Essas fronteiras foram utilizadas na discussão dos resultados. Para os resultados do `auto-sklearn`, utilizamos a média do Macro F-score resultante das cinco execuções dessa estratégia para cada conjunto de dados. Os resultados deste capítulo foram organizados da seguinte forma: a Seção 9.6.1 apresenta o desempenho dos modelos substitutos; a Seção 9.6.2 compara EMANUEL$_{FS}$ e EMANUEL$_{FS+}$ em

relação aos objetivos; e a Seção 9.6.3 compara EMANUEL$_{FS+}$ com os *baselines*.

9.6.1 Desempenho dos Modelos Substitutos

Para verificar o desempenho dos modelos substitutos de EMANUEL$_{FS+}$ utilizamos a primeira fronteira não dominada real. Para essa fronteira, conhecemos os valores reais e previstos para o Macro F-score e para o logaritmo do tamanho do modelo. A Figura 64 ilustra o R^2 dos modelos substitutos para cada conjunto de dados. Quanto mais próximo de um os valores de R^2, mais os modelos substitutos conseguem aproximar suas estimativas aos valores reais. Para o logaritmo do tamanho do modelo, os valores de R^2 foram superiores a 0.87 e para o Macro F-score acima de 0.91. Esses valores indicam que os modelos substitutos tiveram um bom desempenho.

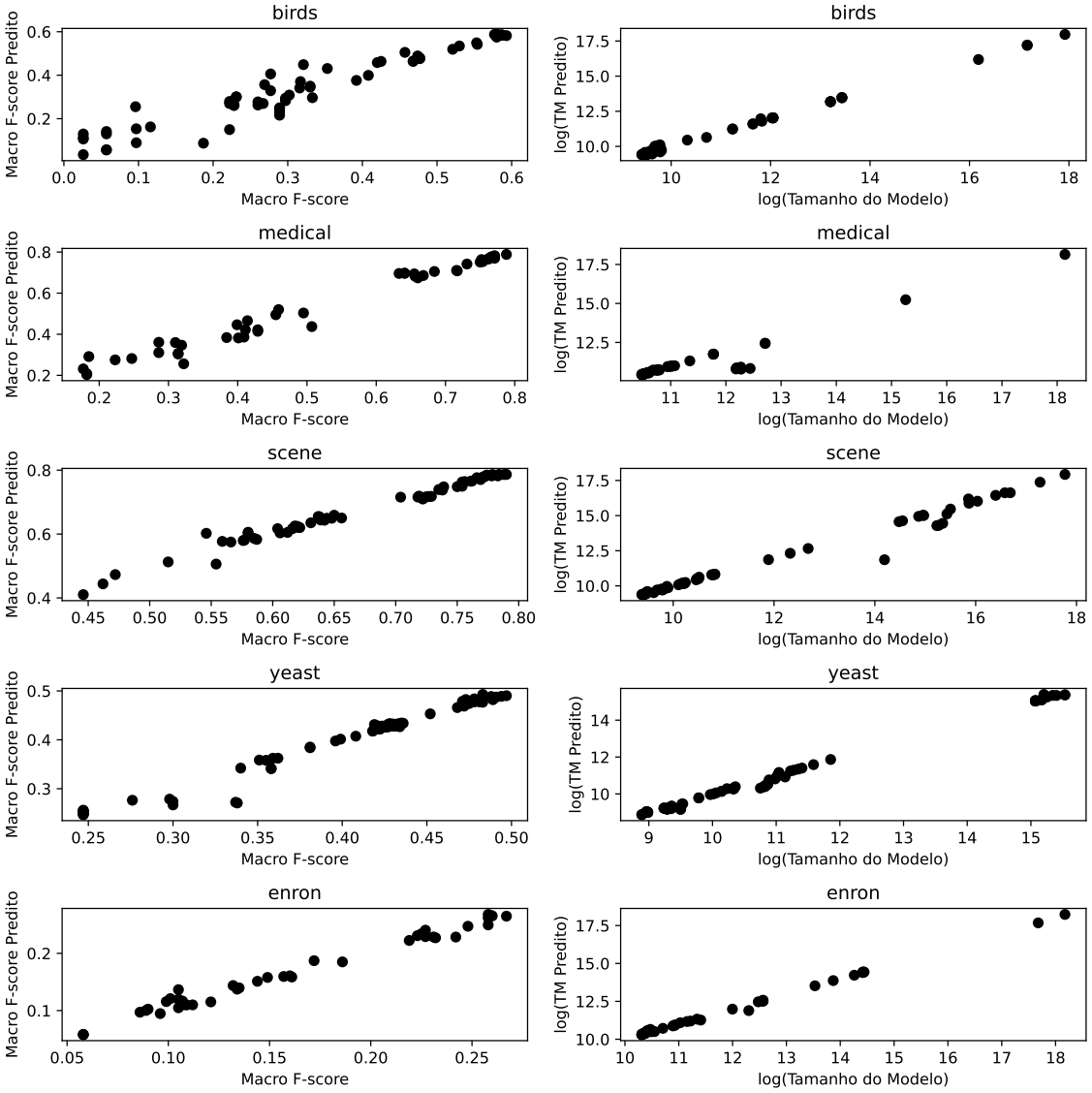
Figura 64 – R^2 dos modelos substitutos que preveem o Macro F-score e o logaritmo do tamanho do modelo.



Além da Figura 64, a Figura 65 apresenta os diagramas de dispersão dos valores reais e previstos pelos modelos substitutos para cada conjunto de dados. Em todos os diagramas, os dados tendem a formar uma reta com inclinação um, indicando que os modelos substitutos fazem previsões precisas. Os pontos acima ou abaixo da reta representam previsões inconsistentes em que o modelo substituto estimou valores superiores ou inferiores aos valores reais, respectivamente.

O menor R^2 para modelos que preveem o Macro F-score ocorreu no conjunto de dados *birds*. Na Figura 65, dentre os diagramas da primeira coluna (onde o atributo-alvo é o Macro F-score), o diagrama desse conjunto de dados é o que apresenta maior dispersão em relação à reta com inclinação um. Já para os modelos que preveem o logaritmo do tamanho do modelo, o pior R^2 ocorreu no conjunto de dados *medical*. Dentre os diagramas da segunda coluna da Figura 65 (onde o atributo-alvo é o logaritmo do tamanho do modelo), o diagrama do conjunto de dados *medical* também está entre os diagramas com as maiores dispersões, o que justifica seu R^2 inferior.

Figura 65 – Diagrama de dispersão para o Macro F-score e para o logaritmo do tamanho do modelo (TM).



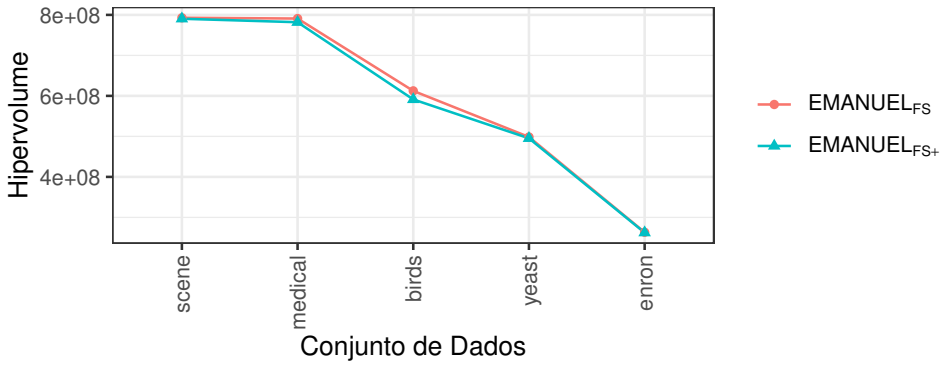
Os resultados demonstram que os modelos substitutos da estratégia EMANUEL$_{FS+}$ apresentaram estimativas próximas aos valores reais, tanto para o Macro F-score quanto para o logaritmo do tamanho do modelo. Mesmo os modelos com os menores valores de R^2 apresentaram coeficientes consideravelmente altos, demonstrando sua boa capacidade preditiva e confirmando a eficácia da estratégia proposta com modelos substitutos.

9.6.2 EMANUEL$_{FS}$ x EMANUEL$_{FS+}$

Como EMANUEL$_{FS+}$ é uma extensão de EMANUEL$_{FS}$ que usa modelos substitutos, as discussões desta seção concentram-se na comparação dessas duas estratégias. Iniciamos a comparação com a análise do hipervolume. A Figura 66 ilustra os hipervolumes

resultantes dessas estratégias por meio de um gráfico de linhas. Ambas as estratégias apresentam hipervolumes muito próximos. No entanto, o hipervolume de EMANUEL$_{FS+}$ é ligeiramente inferior ao de EMANUEL$_{FS}$, o que sugere que EMANUEL$_{FS}$ pode encontrar soluções que otimizam melhor os valores dos objetivos do que EMANUEL$_{FS+}$.

Figura 66 – Hipervolume das fronteiras de Pareto de EMANUEL$_{FS}$ e de EMANUEL$_{FS+}$.



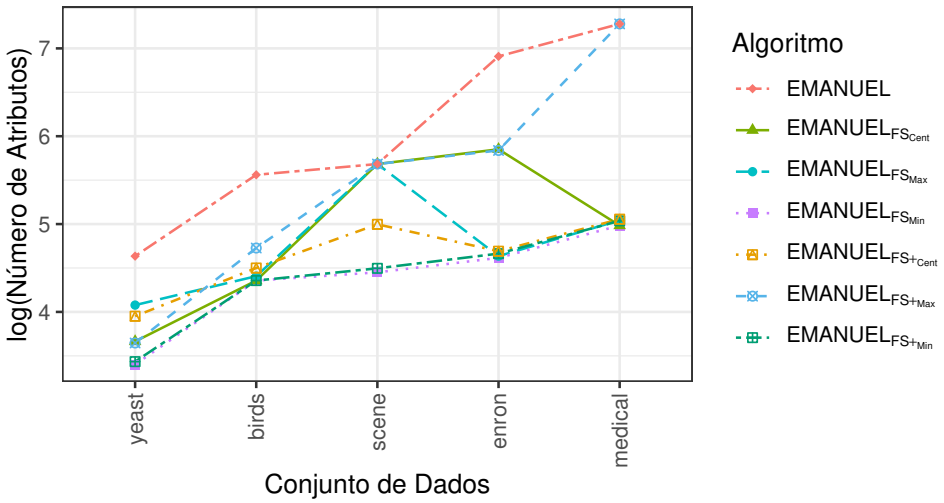
Para avaliar as soluções das fronteiras de EMANUEL$_{FS}$ e de EMANUEL$_{FS+}$ selecionamos soluções com valores mínimos (*Min*), valores intermediários (*Cent*) e valores máximos (*Max*) dos objetivos das fronteiras de Pareto (conforme descrito na Seção 5.1.4). As soluções de EMANUEL$_{FS}$ foram denominadas EMANUEL$_{\text{FS}_{\text{Min}}}$, EMANUEL$_{\text{FS}_{\text{Cent}}}$ e EMANUEL$_{\text{FS}_{\text{Max}}}$. Já as soluções de EMANUEL$_{FS+}$ foram denominadas EMANUEL$_{\text{FS+}_{\text{Min}}}$, EMANUEL$_{\text{FS+}_{\text{Cent}}}$ e EMANUEL$_{\text{FS+}_{\text{Max}}}$. Esperávamos que as soluções *Min* das estratégias tivessem desempenhos semelhantes entre si em relação aos objetivos, assim como as soluções *Cent* e *Max*.

Tanto EMANUEL$_{FS}$ quanto EMANUEL$_{FS+}$ podem fazer a seleção de atributos antes da indução dos modelos de classificação. A seleção de atributos não contribui necessariamente com a melhora nos valores dos objetivos, mas ela contribui para que algoritmos e/ou configurações de hiperparâmetros mais complexos participem do processo evolutivo do NSGA-II, uma vez que esses algoritmos executam com restrições de tempo. A Figura 67 apresenta o número de atributos utilizados pelos algoritmos selecionados das fronteiras de EMANUEL$_{FS}$ e de EMANUEL$_{FS+}$. Além disso, apresenta o limite superior do número de atributos por meio da estratégia EMANUEL, que utiliza todos os atributos dos conjuntos de dados.

Ambas as estratégias poderiam ou não realizar a seleção de atributos. Na Figura 67, os algoritmos que não efetuaram a seleção de atributos foram EMANUEL$_{\text{FS}_{\text{Max}}}$, EMANUEL$_{\text{FS}_{\text{Cent}}}$ e EMANUEL$_{\text{FS+}_{\text{Max}}}$ para o conjunto de dados *scene*, e EMANUEL$_{\text{FS+}_{\text{Max}}}$ para o conjunto de dados *medical*. Os demais algoritmos trabalharam com números de atributos significativamente reduzidos em relação ao número de atributos original dos conjuntos de dados. Para melhorar a visualização desses resultados, a Tabela 14 traz o

número de atributos dos conjuntos de dados e o número de atributos utilizados pelos algoritmos das estratégias.

Figura 67 – Número de atributos utilizados pelos algoritmos selecionados das fronteiras de EMANUEL$_{FS}$ e de EMANUEL$_{FS+}$. A estratégia EMANUEL utiliza todos os atributos dos conjuntos de dados e foi incluída no gráfico para representar o limite superior do número de atributos.



A Figura 68 mostra o Macro F-score dos algoritmos selecionados das fronteiras das estratégias avaliadas. No gráfico de linhas da figura, as linhas dos algoritmos com valores mínimos dos objetivos (*Min*) têm comportamentos similares. O mesmo ocorre com as linhas do algoritmos com valores intermediários (*Cent*) e com valores máximos (*Max*) de Macro F-score. Além disso, essas linhas estão muito próximas entre si, o que indica que os Macro F-scores desses algoritmos são similares. Para facilitar a interpretação da Figura 68, a Figura 69 classifica os algoritmos selecionados das fronteiras de EMANUEL$_{FS}$ e de EMANUEL$_{FS+}$ de acordo com os seus Macro F-score, em que classificações menores (quadrados azuis) correspondem aos melhores algoritmos.

Na Figura 69 as melhores classificações são as soluções *Max*, *Cent* e *Min* de ambas as estratégias. EMANUEL$_{\text{FS}_{\text{Max}}}$ supera EMANUEL$_{\text{FS+}_{\text{Max}}}$ em todos os conjuntos de dados, enquanto EMANUEL$_{\text{FS}_{\text{Cent}}}$ tem desempenho superior a EMANUEL$_{\text{FS+}_{\text{Cent}}}$ em três deles. Já EMANUEL$_{\text{FS+}_{\text{Min}}}$ possui desempenho igual ou superior a EMANUEL$_{\text{FS}_{\text{Min}}}$ na maioria dos conjuntos de dados. Embora os algoritmos de ambas as estratégias alternem entre os melhores e os piores Macro F-score, as diferenças de desempenho entre os pares de algoritmos de ambas as estratégias (*Min* e *Min*, *Cent* e *Cent*, *Max* e *Max*) são muito pequenas, como podemos observar na Tabela 14. Essa proximidade dos resultados sugere que as estratégias avaliadas não apresentaram diferenças estatísticas significativas em relação ao Macro F-score.

Considerando o tamanho dos modelos, a Figura 70 apresenta os tamanhos dos

Tabela 14 – Resultados de EMANUEL$_{FS}$ e de EMANUEL$_{FS+}$ para o número de atributos (atrib.), o Macro F-score (Mac. F1) e o tamanho dos modelos (T. Modelos). Na tabela destacamos os melhores resultados para o Macro F-score e o tamanho dos modelos, para cada solução selecionada das fronteiras de Pareto resultantes das estratégias.

Conj. de Dados	Solução	Atrib. Total	EMANUEL$_{FS}$			EMANUEL$_{FS+}$		
			Atrib.	Macro F1	T. Modelo	Atrib.	Macro F1	T. Modelo
birds	Min	260	78	0.0570	1.24e+04	78	0.0570	1.22e+04
	Cent		78	0.5967	5.15e+05	90	0.5887	5.48e+05
	Max		82	0.6127	4.75e+07	113	0.5913	1.06e+07
enron	Min	1001	101	0.0920	2.88e+04	106	0.0980	3.01e+04
	Cent		347	0.2623	1.89e+06	109	0.2253	1.62e+05
	Max		103	0.2680	2.20e+09	343	0.2623	1.86e+06
medical	Min	1449	146	0.1823	3.32e+04	154	0.2197	3.54e+04
	Cent		146	0.7517	6.31e+04	157	0.7683	1.31e+05
	Max		155	0.7910	1.29e+05	1449	0.7830	7.56e+07
scene	Min	294	85	0.4853	1.24e+04	89	0.4453	1.19e+04
	Cent		294	0.7607	9.61e+04	148	0.7567	2.25e+05
	Max		294	0.7930	1.44e+07	294	0.7907	1.76e+07
yeast	Min	103	30	0.2620	7.76e+03	31	0.2620	7.85e+03
	Cent		39	0.4627	1.08e+05	52	0.4777	1.41e+05
	Max		59	0.4987	6.40e+06	38	0.4953	5.55e+06

Figura 68 – Macro F-scores dos modelos induzidos pelos algoritmos selecionados das fronteiras de EMANUEL$_{FS}$ e de EMANUEL$_{FS+}$.

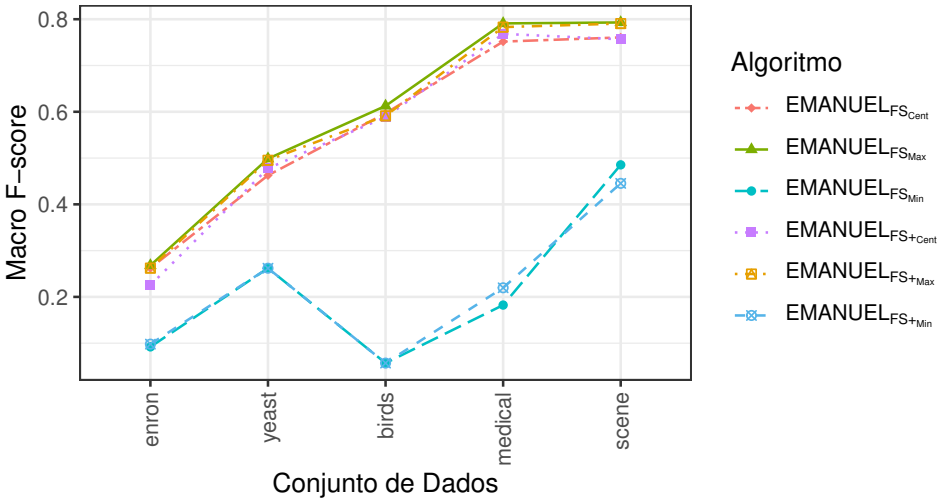
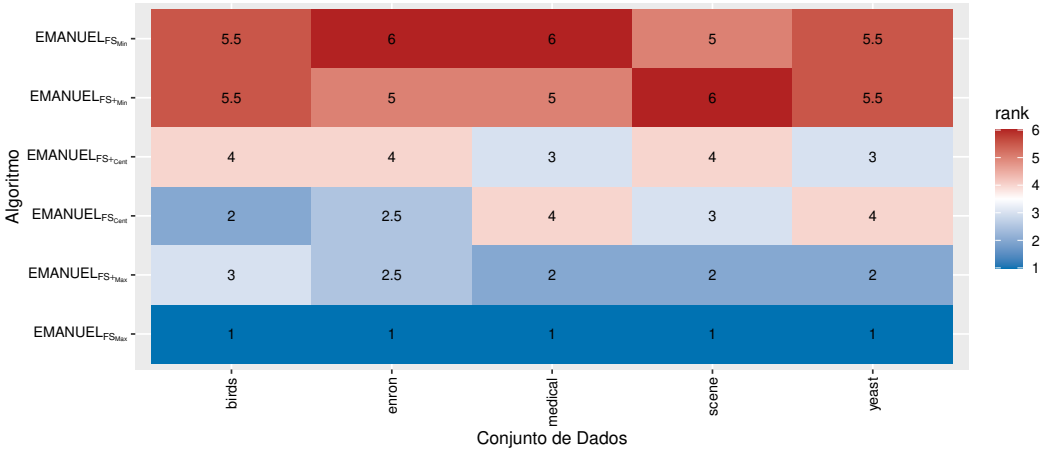
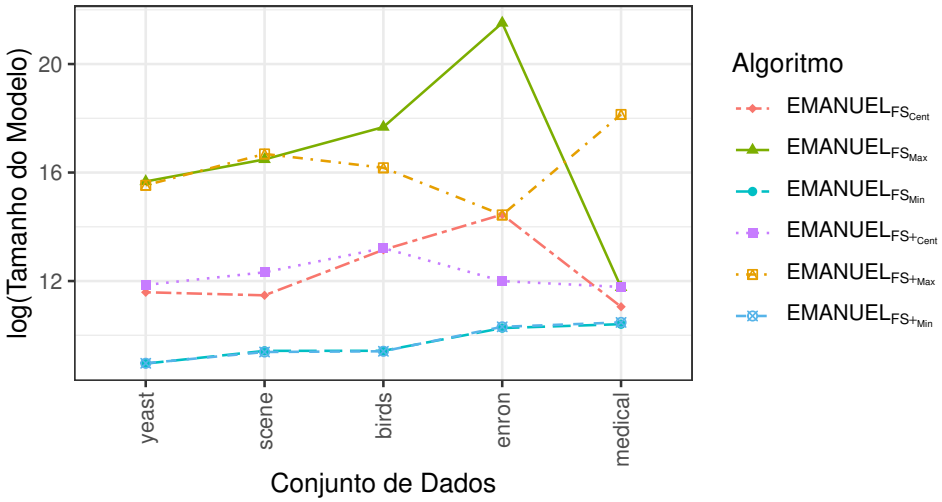


Figura 69 – *Ranking* dos algoritmos selecionados das fronteiras de EMANUEL$_{FS}$ e de EMANUEL$_{FS+}$ em relação ao Macro F-score.



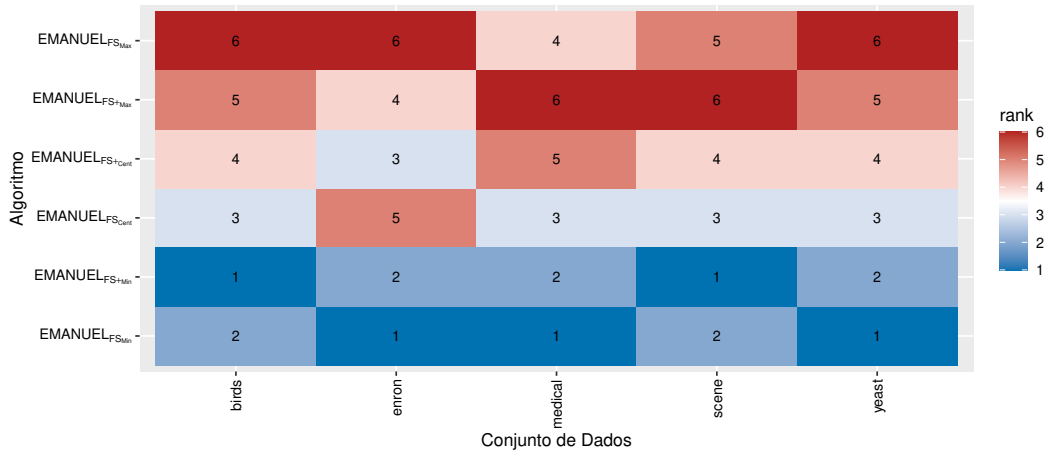
modelos induzidos pelos algoritmos selecionados das fronteiras de EMANUEL$_{FS}$ e de EMANUEL$_{FS+}$. Diferentemente da Figura 68, na Figura 70, as linhas dos algoritmos com valores intermediários (EMANUEL$_{\text{FS}_{\text{Cent}}}$ e EMANUEL$_{\text{FS+}_{\text{Cent}}}$) e máximos (EMANUEL$_{\text{FS}_{\text{Max}}}$ e EMANUEL$_{\text{FS+}_{\text{Max}}}$) do tamanho dos modelos nem sempre têm comportamentos similares. Esse fato pode ser justificado pelas diferentes opções de algoritmos e hiperparâmetros disponíveis nos hiperespaços para a seleção de atributos e para a indução dos modelos de classificação.

Figura 70 – Logaritmo do tamanho dos modelos induzidos pelos algoritmos selecionados das fronteiras de EMANUEL$_{FS}$ e de EMANUEL$_{FS+}$.



Em complemento à Figura 70, a Figura 71 classifica os algoritmos selecionados das fronteiras de EMANUEL$_{FS}$ e de EMANUEL$_{FS+}$ de acordo com os seus tamanhos de modelos. EMANUEL$_{\text{FS}_{\text{Min}}}$ constrói modelos menores que EMANUEL$_{\text{FS+}_{\text{Min}}}$ em três conjun-

Figura 71 – *Ranking* dos algoritmos selecionados das fronteiras de EMANUEL$_{FS}$ e de EMANUEL$_{FS+}$ em relação ao tamanho do modelo

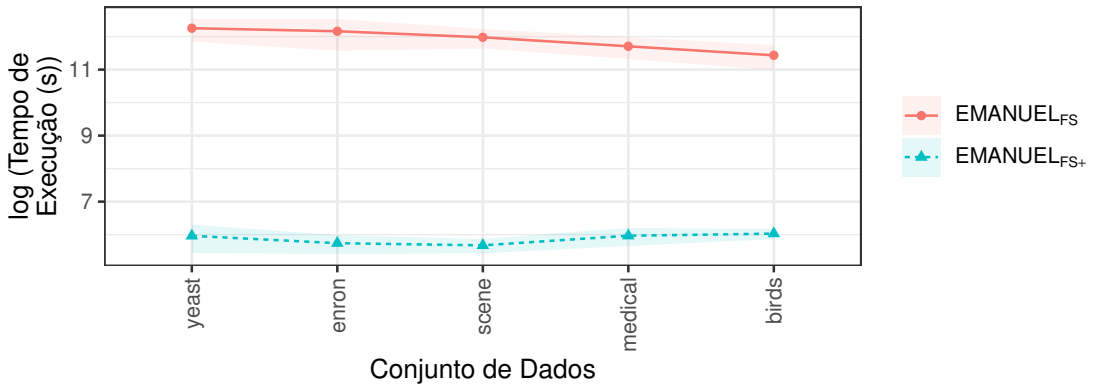


tos de dados. EMANUEL$_{\text{FS}_{\text{Cent}}}$ induz modelos menores que EMANUEL$_{\text{FS+}_{\text{Cent}}}$ em quatro deles. Já os modelos de EMANUEL$_{\text{FS}_{\text{Max}}}$ são menores que os de EMANUEL$_{\text{FS+}_{\text{Max}}}$ em dois conjuntos de dados. Assim, as soluções de EMANUEL$_{FS}$ e EMANUEL$_{FS+}$ treinam modelos menores ou maiores a depender do conjunto de dados e as diferenças dos tamanhos dos modelos entre os pares de algoritmos de ambas as estratégias (*Min* e *Min*, *Cent* e *Cent*, *Max* e *Max*) podem ser elevadas, como mostra a Tabela 14.

Para verificar se EMANUEL$_{FS}$ e EMANUEL$_{FS+}$ apresentavam diferenças estatísticas significativas em relação ao Macro F-score e ao tamanho dos modelos, aplicamos o teste estatístico de Wilcoxon. Nos testes, utilizamos os pares de soluções selecionadas das fronteiras das duas estratégias. A hipótese nula foi aceita no teste de Wilcoxon para o Macro F-score (*p-value = 0.422*) e no teste de Wilcoxon para o tamanho do modelo (*p-value = 0.762*), indicando que não há diferenças significativas entre os resultados das estratégias avaliadas. Portanto, podemos concluir que as estimativas dos modelos substitutos para o Macro F-score e o tamanho do modelo são suficientemente precisas, permitindo que EMANUEL$_{FS+}$ alcance resultados equivalentes aos de EMANUEL$_{FS}$.

A última análise é em relação ao tempo de execução das estratégias EMANUEL$_{FS}$ e EMANUEL$_{FS+}$. A Figura 72 ilustra o tempo médio das cinco execuções das estratégias. EMANUEL$_{FS}$ apresenta tempo de execução significativamente superior ao de EMANUEL$_{FS+}$. Essa diferença deve-se às características de cada uma das estratégias. EMANUEL$_{FS}$ faz a seleção de atributos e a indução de um modelo de classificação multirrótulo todas as vezes que uma solução candidata é avaliada. Já EMANUEL$_{FS+}$ utiliza as predições dos modelos substitutos para avaliar uma solução candidata, o que é muito mais rápido de executar. Dessa forma, EMANUEL$_{FS+}$ beneficia-se da agilidade dos modelos substitutos de estimar o desempenho das soluções candidatas.

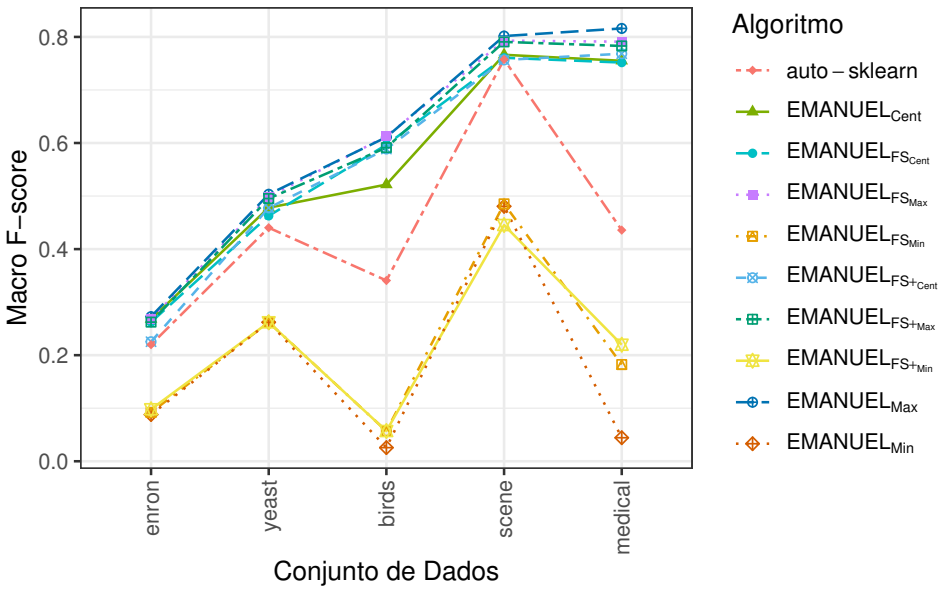
Figura 72 – Logaritmo do tempo de execução das estratégias EMANUEL$_{FS}$ e EMANUEL$_{FS+}$.



9.6.3 Comparação com os Algoritmos *Baselines*

Nesta seção comparamos EMANUEL$_{FS+}$ com as estratégias EMANUEL, EMANUEL$_{FS}$ e `auto-sklearn`. Para a análise dos resultados, utilizamos as três soluções selecionadas das fronteiras de EMANUEL (EMANUEL$_{\text{Min}}$, EMANUEL$_{\text{Cent}}$ e EMANUEL$_{\text{Max}}$), EMANUEL$_{FS}$ e EMANUEL$_{FS+}$. No caso do `auto-sklearn`, utilizamos apenas a média do Macro F-score (resultante das cinco execuções), pois não há informações disponíveis sobre o tamanho dos modelos gerados por essa ferramenta.

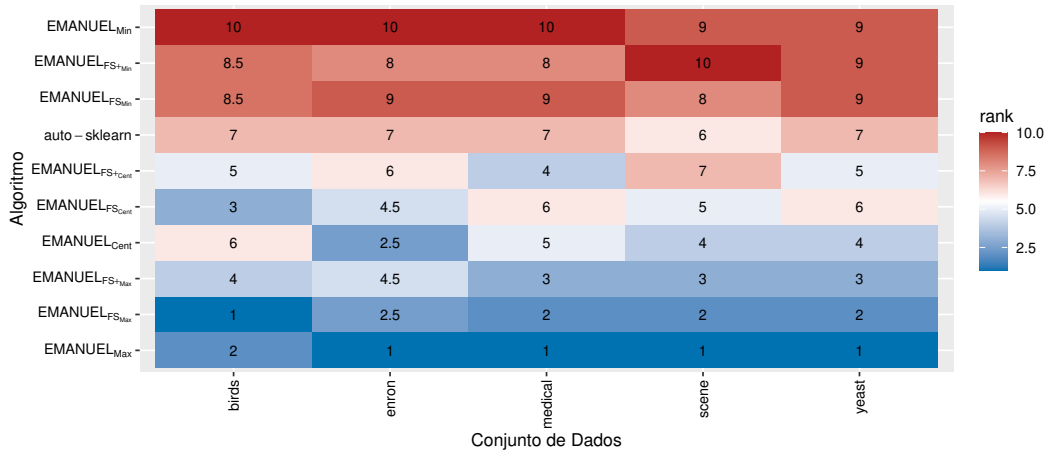
Figura 73 – Macro F-scores dos algoritmos avaliados no experimento.



A Figura 73 mostra os resultados dos algoritmos para o Macro F-score. No gráfico, os algoritmos EMANUEL$_{\text{Min}}$, EMANUEL$_{\text{FS}_{\text{Min}}}$ e EMANUEL$_{\text{FS+}_{\text{Min}}}$ apresentam os me-

nores Macro F-scores. Em seguida, temos o `auto-sklearn` e, por fim, as soluções com Macro F-scores intermediários (*Cent*) e máximos (*Max*) das fronteiras de EMANUEL, EMANUEL$_{FS}$ e EMANUEL$_{FS+}$. As linhas com os maiores valores do Macro F-score se sobrepõem no gráfico da Figura 73. Para ajudar a visualizar essas diferenças, a Figura 74 classifica os algoritmos em relação ao Macro F-score. Na figura, os *rankings* menores representam os algoritmos com os melhores Macro F-scores. Os algoritmos estão dispostos pelo *ranking* médio. Considerando o *ranking* médio, EMANUEL$_{\text{Max}}$, EMANUEL$_{\text{FS}_{\text{Max}}}$, EMANUEL$_{\text{FS+}_{\text{Max}}}$, EMANUEL$_{\text{Cent}}$, EMANUEL$_{\text{FS}_{\text{Cent}}}$ e EMANUEL$_{\text{FS+}_{\text{Cent}}}$ são os algoritmos com os melhores Macro F-scores.

Figura 74 – *Ranking* dos algoritmos avaliados no experimento em relação ao Macro F-score.



A Figura 74 classifica os algoritmos, mas quais são as diferenças entre os valores de Macro F-score desses algoritmos? A Tabela 15 apresenta os valores numéricos correspondentes aos utilizados nas Figuras 73 e 74, com os melhores valores destacados. EMANUEL$_{\text{Max}}$ obteve os melhores resultados para o Macro F-score na maioria dos conjuntos de dados. No entanto, essa vantagem não foi tão expressiva quando comparamos EMANUEL$_{\text{Max}}$ com EMANUEL$_{\text{FS}_{\text{Max}}}$ e EMANUEL$_{\text{FS+}_{\text{Max}}}$. O mesmo ocorreu com os algoritmos selecionados do centro das fronteiras de Pareto: EMANUEL$_{\text{Cent}}$ obteve Macro F-score superior aos de EMANUEL$_{\text{FS}_{\text{Cent}}}$ e EMANUEL$_{\text{FS+}_{\text{Cent}}}$ na maioria dos conjuntos de dados, mas as diferenças de desempenho não foram significativas.

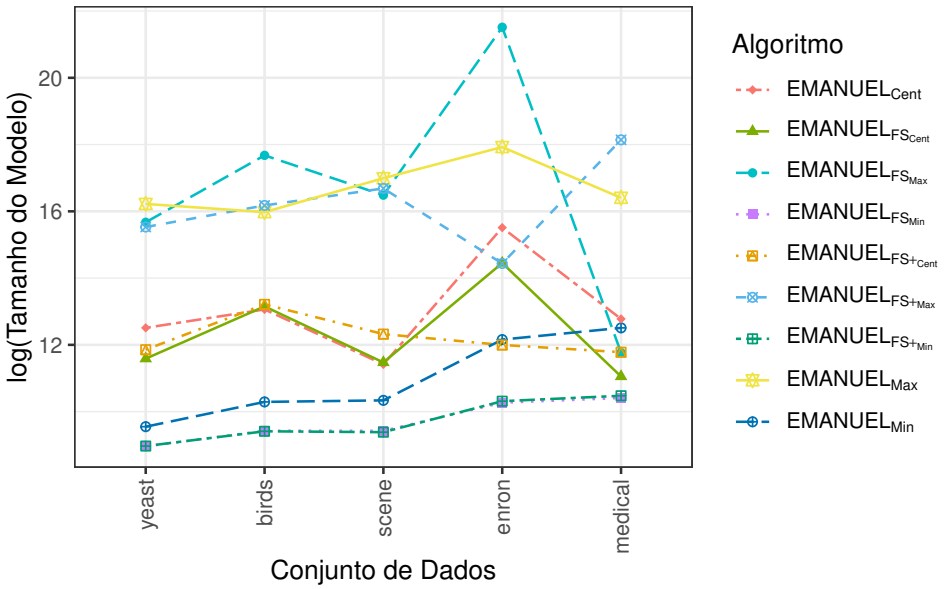
Nas comparações específicas entre EMANUEL$_{\text{Max}}$ e `auto-sklearn`, há casos em que o EMANUEL$_{\text{Max}}$ obteve melhorias significativas, como nos conjuntos de dados *birds* (0.611 vs. 0.341) e *medical* (0.816 vs. 0.436). Nesses conjuntos de dados, EMANUEL$_{\text{FS}_{\text{Max}}}$, EMANUEL$_{\text{FS+}_{\text{Max}}}$, EMANUEL$_{\text{Cent}}$, EMANUEL$_{\text{FS}_{\text{Cent}}}$ e EMANUEL$_{\text{FS+}_{\text{Cent}}}$ também superaram o `auto-sklearn`. Por outro lado, o `auto-sklearn` nunca obteve uma vantagem tão expressiva contra esses algoritmos.

Tabela 15 – Macro F-score dos algoritmos avaliados no experimento. Na tabela destacamos os melhores resultados para o Macro F-score.

Conj.de Dados	**EMANUEL**			**EMANAUEL$_{FS}$**			**EMANUEL$_{FS+}$**			**auto-sklearn**
	Min	**Cent**	**Max**	**Min**	**Cent**	**Max**	**Min**	**Cent**	**Max**	
birds	0.026	0.522	0.611	0.057	0.597	0.613	0.057	0.589	0.591	0.341
enron	0.088	0.268	0.273	0.092	0.262	0.268	0.098	0.220	0.262	0.220
medical	0.044	0.755	0.816	0.182	0.752	0.791	0.220	0.768	0.783	0.436
scene	0.481	0.766	0.802	0.485	0.761	0.793	0.445	0.757	0.791	0.758
yeast	0.262	0.478	0.503	0.262	0.463	0.499	0.262	0.478	0.495	0.440

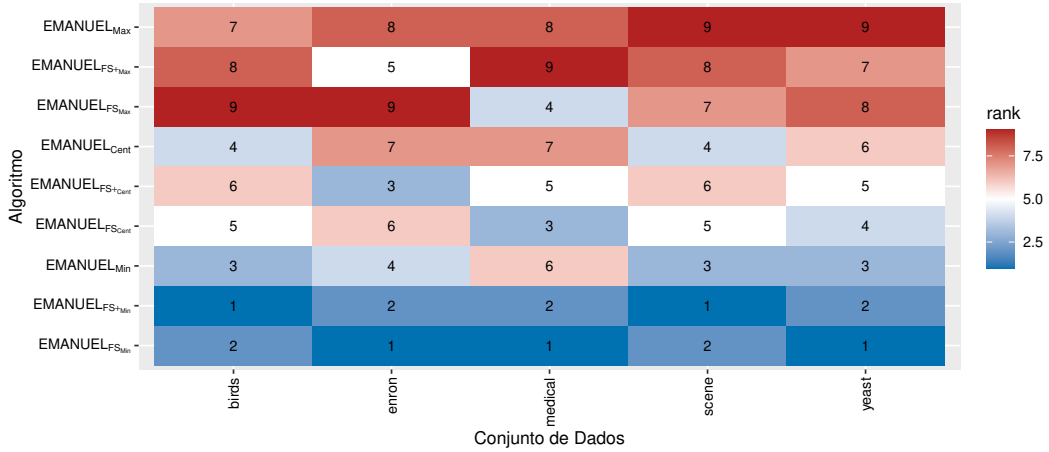
A Figura 75 apresenta o tamanho dos modelos induzidos com os algoritmos selecionados das fronteiras de Pareto resultantes das estratégias EMANUEL, EMANUEL$_{FS}$ e EMANUEL$_{FS+}$. Conforme mencionamos, não foi possível realizar essa análise com o `auto-sklearn`, pois não temos essa informação para essa ferramenta. No gráfico, os algoritmos EMANUEL$_{Min}$, EMANUEL$_{\text{FS}_{\text{Min}}}$ e EMANUEL$_{\text{FS+}_{\text{Min}}}$ apresentaram os menores tamanhos dos modelos na maioria dos conjuntos de dados. Em seguida, estão os algoritmos selecionados das fronteiras de Pareto com tamanhos de modelos intermediários (*Cent*) e máximos (*Max*).

Figura 75 – Logaritmo do tamanho dos modelos induzidos pelos algoritmos avaliados no experimento.



Para melhor visualização das diferenças de tamanhos dos modelos, a Figura 76 classifica os algoritmos em relação a essa medida. Considerando o *ranking* médio, EMANUEL$_{\text{FS}_{\text{Min}}}$, EMANUEL$_{\text{FS+}_{\text{Min}}}$, EMANUEL$_{Min}$, EMANUEL$_{\text{FS}_{\text{Cent}}}$, EMANUEL$_{\text{FS+}_{\text{Cent}}}$ e EMANUEL$_{Cent}$, são os algoritmos com os melhores tamanhos de modelos. A Tabela

Figura 76 – *Ranking* dos algoritmos avaliados no experimento em relação ao tamanho dos modelos.



16 apresenta os valores numéricos dos tamanhos dos modelos utilizados nas Figuras 75 e 76, com os melhores valores destacados. Dentre os algoritmos *Min*, os que realizam seleção de atributos apresentaram resultados melhores quanto ao tamanho dos modelos na maioria das vezes. O mesmo ocorreu com os algoritmos *Cent*, onde EMANUEL$_{\text{FS}_{\text{Cent}}}$ e EMANUEL$_{\text{FS+Cent}}$ apresentam resultados melhores do que EMANUEL$_{Cent}$ na maioria dos conjuntos de dados. Diferente do que ocorreu com o Macro F-score, as variações no tamanho dos modelos apresentaram discrepâncias significativas.

Tabela 16 – Tamanhos dos modelos induzidos pelos algoritmos das estratégias EMANUEL, EMANUEL$_{FS}$ e EMANUEL$_{FS+}$. Na tabela destacamos os melhores resultados quanto a essa medida.

Conj. Dados	**EMANUEL**			**EMANAUEL$_{\text{FS}}$**			**EMANUEL$_{\text{FS+}}$**		
	Min	**Cent**	**Max**	**Min**	**Cent**	**Max**	**Min**	**Cent**	**Max**
birds	2.95e+04	4.70e+05	8.68e+06	1.24e+04	5.15e+05	4.75e+07	1.22e+04	5.48e+05	1.06e+07
enron	1.91e+05	5.45e+06	6.09e+07	2.88e+04	1.89e+06	2.20e+09	3.01e+04	1.62e+05	1.86e+06
medical	2.71e+05	3.54e+05	1.33e+07	3.32e+04	6.31e+04	1.29e+05	3.54e+04	1.31e+05	7.56e+07
scene	3.09e+04	9.07e+04	2.40e+07	1.24e+04	9.61e+04	1.44e+07	1.19e+04	2.25e+05	1.76e+07
yeast	1.40e+04	2.71e+05	1.11e+07	7.76e+03	1.08e+05	6.40e+06	7.85e+03	1.41e+05	5.55e+06

Assim, EMANUEL$_{\text{Max}}$, EMANUEL$_{\text{FS}_{\text{Max}}}$ e EMANUEL$_{\text{FS+}_{\text{Max}}}$ apresentaram os melhores *rankings* médios para o Macro F-score (Figura 74). Para o tamanho do modelo, os melhores *rankings* médios foram alcançados por EMANUEL$_{\text{FS}_{\text{Min}}}$, EMANUEL$_{\text{FS+}_{\text{Min}}}$ e EMANUEL$_{Min}$ (Figura 76). No entanto, não foi possível verificar a existência de diferenças estatisticamente significativas entre esses algoritmos e os demais algoritmos avaliados no experimento devido ao número insuficiente de conjuntos de dados para a realização do teste estatístico.

9.7 Considerações Finais

Neste capítulo propomos EMANUEL$_{FS+}$, uma extensão de EMANUEL$_{FS}$, que utiliza modelos substitutos para prever os valores dos objetivos. Como prova de conceito, o experimento foi conduzido utilizando apenas cinco conjuntos de dados. Os modelos substitutos alcançaram alta precisão preditiva, com coeficientes de determinação elevados, superiores a 0.87 para as estimativas do logaritmo do tamanho dos modelos e acima de 0.91 para o Macro F-score. Esses resultados validam a qualidade das estimativas feitas pelos modelos substitutos, utilizados na estratégia proposta.

Comparamos EMANUEL$_{FS}$ e EMANUEL$_{FS+}$ para validar a hipótese de pesquisa. Os testes estatísticos não identificaram diferenças estatisticamente significativas entre os Macro F-scores nem entre os tamanhos de modelos resultantes das duas estratégias, confirmando que a inclusão dos modelos substitutos em EMANUEL$_{FS+}$ não compromete seus resultados finais. No entanto, EMANUEL$_{FS+}$ diminuiu significativamente o tempo de execução em relação à estratégia EMANUEL$_{FS}$. Esses resultados confirmam a validade da hipótese de pesquisa.

Estendemos as comparações da estratégia proposta, incluindo `auto-sklearn` e EMANUEL como algoritmos *baselines*. Os algoritmos selecionados da fronteira de Pareto resultante de EMANUEL$_{FS+}$ estiveram entre os melhores resultados tanto para o Macro F-score quanto para o tamanho do modelo (considerando o *ranking* médio). No entanto, não foi possível verificar a existência de diferenças estatisticamente significativas entre EMANUEL$_{FS+}$ e os demais *baselines* devido ao número limitado de conjuntos de dados.

Até onde sabemos, EMANUEL$_{FS+}$ é a primeira estratégia AutoML para classificação multirrótulo que automatiza a seleção de atributos e a busca pelo algoritmo de classificação, utilizando meta-aprendizado para contribuir com a execução. Os resultados alcançados foram promissores e suficientes para esta prova de conceito. No entanto, para fortalecer os resultados já encontrados, seria interessante conduzir novos experimentos incluindo novos conjuntos de dados e adicionando outras estratégias AutoML de última geração como *baselines*. A ampliação do escopo experimental permitiria avaliar com maior robustez a estratégia EMANUEL$_{FS+}$.

Capítulo 10

Conclusão

Nesta tese, investigamos o uso do AutoML para a classificação multirrótulo, propondo novas estratégias AutoML com requisitos relacionados à otimização multiobjetivo, seleção de atributos e meta-aprendizado. As novas estratégias foram desenvolvidas e validadas por meio de experimentos a fim de obter resultados empíricos que permitissem responder às questões de pesquisa definidas no Capítulo 1. Organizamos este capítulo respondendo às questões de pesquisa e identificando limitações e trabalhos futuros.

10.1 Respondendo as Questões de Pesquisa

É viável e eficaz empregar otimização multiobjetivo em estratégias AutoML para classificação multirrótulo?

As estratégias AutoML propostas otimizam simultaneamente o Macro F-score e o tamanho dos modelos induzidos. O resultado de uma estratégia, para um conjunto de dados multirrótulo, é a fronteira de Pareto contendo soluções com diferentes valores para os objetivos. Um usuário pode escolher dessa fronteira a solução que atenda seus requisitos de desempenho e/ou de complexidade do modelo. Escolhemos soluções nos extremos da fronteira, com valores mínimos (*Min*) e máximos (*Max*) para os objetivos. Também escolhemos soluções no centro das fronteiras (*Cent*).

As soluções *Min* representam algoritmos que caracterizam-se por induzirem modelos pequenos, mas que possuem Macro F-score incipiente. As soluções *Max* apresentam Macro F-score elevados e os modelos induzidos têm tamanhos elevados. Já as soluções *Cent* tendem a otimizar ambos os objetivos. Dessa forma, podemos avaliar os resultados

das estratégias EMANUEL, EMANUEL$_{SM}$, EMANUEL$_{FS}$ e EMANUEL$_{FS+}$ (Seções 5.3, 6.6.3, 8.6 e 9.6.3, respectivamente) a depender da escolha do usuário:

- ❏ A escolha são modelos de tamanhos pequenos: as soluções *Min* das estratégias propostas pertencem ao grupo de algoritmos com tamanhos de modelos estatisticamente menores.
- ❏ A escolha são modelos que ponderam ambos os objetivos: as soluções *Cent* das estratégias supracitadas pertencem aos grupos de algoritmos com Macro F-score estatisticamente superiores e tamanhos de modelos estatisticamente inferiores.
- ❏ A escolha são modelos com Macro F-score elevados: as soluções *Max* das estratégias supracitadas pertencem ao grupo de algoritmos com Macro F-score estatisticamente maiores.

Esses resultados e conclusões indicam que as estratégias EMEMANUEL, EMANUEL$_{SM}$, EMANUEL$_{FS}$ e EMANUEL$_{FS+}$ geram um conjunto diversificado de soluções não dominadas, que proporcionam maior flexibilidade para a seleção de classificadores que mantêm desempenho competitivo, comparável a outros algoritmos. Embora esses resultados sejam característicos da otimização multiobjetivo e também possam ser observáveis na classificação monorrótulo, sua contribuição é particularmente relevante na classificação multirrótulo. Nesse contexto, o hiperespaço contém algoritmos de diferentes abordagens, apresenta elevada complexidade computacional e envolve conflitos naturais entre as próprias métricas de avaliação e entre as estruturas dos modelos induzidos. Dessa forma, a otimização multiobjetivo pode capturar relações de compromisso que são inerentes à classificação multirrótulo e encontrar os modelos mais adequados às diferentes restrições práticas. Além disso, a adoção da otimização multiobjetivo em estratégias AutoML específicas para classificação multirrótulo ainda não havia sido explorada na literatura, o que reforça a originalidade e a relevância desta contribuição.

Como reduzir o tempo de execução do AutoML sem comprometer a qualidade dos modelos?

No geral, estratégias AutoML tendem a apresentar tempo de execução elevado, pois elas avaliam diretamente vários algoritmos em busca de uma solução ou de um conjunto de soluções que otimizam os objetivos. Nesta pesquisa, utilizamos modelos substitutos nas estratégias AutoML propostas para prever os valores do Macro F-score e do logaritmo do tamanho dos modelos, com o objetivo de reduzir o tempo de execução do AutoML.

Na estratégia EMANUEL$_{SM}$, os modelos substitutos que preveem o logaritmo do tamanho dos modelos e o Macro F-score apresentaram R^2 acima de 0.95 e 0.92, respectivamente. A exceção a esses índices ocorreu para os modelos que preveem o Macro F-score

nos conjuntos de dados *birds* e *mediamill*, onde os valores de R^2 foram inferiores (0.70 e 0.56, respectivamente). Na estratégia EMANUEL$_{FS+}$, os valores de R^2 dos modelos substitutos para o logaritmo do tamanho do modelo foram superiores a 0.87 e para o Macro F-score ficaram acima de 0.91. Considerando esses valores de R^2, os modelos substitutos conseguem fornecer uma boa estimativa do tamanho e do desempenho.

Comparamos estratégias AutoML que integram e que não integram modelos substitutos quanto aos valores dos objetivos. EMANUEL e EMANUEL$_{SM}$ encontram algoritmos que possuem Macro F-scores similares do ponto de vista estatístico. Entretanto, o tamanho dos modelos induzidos por esses algoritmos é estatisticamente diferente. Na maioria dos casos, os modelos induzidos por EMANUEL são maiores que os induzidos por EMANUEL$_{SM}$ (Seção 6.6.2). Também comparamos EMANUEL$_{FS}$ e EMANUEL$_{FS+}$ quanto aos valores dos objetivos. Para essas estratégias, os algoritmos não apresentam diferenças estatisticamente significativas para Macro F-scores e para o tamanho dos modelos induzidos (Seção 9.6.2).

As estratégias EMANUEL$_{SM}$ e EMANUEL$_{FS+}$ demonstraram uma redução significativa no tempo de execução ao integrarem os modelos substitutos (conforme discutido nas Seções 6.6.2 e 9.6.2). Isso ocorreu porque as estratégias deixaram de treinar diretamente os classificadores multirrótulo[1] e passaram a utilizar as previsões dos modelos substitutos. Embora tenhamos treinado os algoritmos das fronteiras de Pareto resultantes para a avaliação das estratégias com valores reais, esse tempo adicional não foi incorporado ao tempo total das estratégias. Vale ressaltar que o número de algoritmos executados para avaliação é substancialmente menor que o volume de algoritmos processados pelo AutoML. Mesmo que o tempo adicional fosse contabilizado no tempo total das estratégias, as estratégias baseadas em modelos substitutos ainda teriam tempo de execução inferior.

No geral, esses resultados e conclusões indicam que as estratégias AutoML para classificação multirrótulo, que empregam modelos substitutos para prever o desempenho dos classificadores (EMANUEL$_{SM}$ e EMANUEL$_{FS+}$), são capazes de manter a qualidade preditiva dos modelos finais, enquanto reduzem o tempo total de execução do AutoML. Mesmo que EMANUEL$_{SM}$ e EMANUEL$_{FS+}$ selecionem classificadores com desempenho ligeiramente inferior, essa perda é compensada pelo menor tempo de resposta proporcionado pela estratégia baseada em modelos substitutos e pela eliminação do trabalho manual de seleção de algoritmo de classificação multirrótulo e configuração de hiperparâmetros.

É possível automatizar a engenharia de atributos, em particular a seleção de atributos, sem comprometer a qualidade preditiva das soluções AutoML?

Para responder a essa questão, conduzimos um experimento preliminar onde propomos AutoMLFS. O objetivo era identificar os algoritmos de seleção de atributos multirrótulo

[1] Em EMANUEL$_{FS+}$, além da indução dos modelos, pode ocorrer também a seleção de atributos.

disponíveis e o efeito da seleção de atributos automatizada no classificador ECC. Comparamos o desempenho de AutoMLFS com modelos ECC induzidos com e sem a seleção de atributos utilizando diferentes algoritmos de seleção. Na maioria dos conjuntos de dados, AutoMLFS encontrou soluções com um número menor de atributos e Macro F-score mais alto em comparação com os *baselines*, mantendo um desempenho estatisticamente equivalente a outros *baselines* (Seção 7.6).

A partir desse experimento, a estratégia EMANUEL$_{FS}$ passou a incluir em seu hiperespaço algoritmos de seleção de atributos multirrótulo. Dessa forma, foi possível realizar a seleção de atributos antes da indução do modelo de classificação multirrótulo. Comparamos EMANUEL e EMANUEL$_{FS}$ quanto aos objetivos. Na maioria dos casos, EMANUEL$_{FS}$ reduziu o número de atributos. O teste de Wilcoxon indicou que não há diferenças significativas entre os resultados, tanto para o Macro F-score quanto para o tamanho dos modelos (Seção 8.6).

Observamos que EMANUEL$_{FS}$ reduz a dimensionalidade dos conjuntos de dados na maioria dos casos. A seleção de atributos pode contribuir ou não para o Macro F-score. Da mesma forma, pode diminuir ou não o tamanho dos modelos. Isso ocorre porque trabalhamos com diferentes algoritmos e configurações de hiperparâmetros. A redução da dimensionalidade pode permitir que algoritmos ou configurações de hiperparâmetros mais complexos participem do processo evolutivo do NSGA-II, considerando os limites de tempo de execução pré-definidos.

Essas conclusões e resultados apresentados demonstram que estratégias AutoML que incorporam a seleção automatizada de atributos são capazes de encontrar soluções que mantêm ou até melhoram o desempenho preditivo em relação a outros algoritmos ou estratégias AutoML que não incorporam essa funcionalidade. Além disso, estas estratégias representam a primeira aplicação de seleção automatizada de atributos em AutoML para classificação multirrótulo, destacando sua originalidade e importância.

10.2 Dificuldades e Limitações

A maior limitação encontrada na tese foi em relação ao tempo de execução. A estratégia EMANUEL levou um tempo considerável para finalizar suas execuções, mesmo utilizando abordagens que pudessem contribuir com a execução, como limitar o tempo de execução para a indução dos modelos, consultar as soluções já avaliadas e implementar a parada antecipada. Essa dificuldade foi potencializada na estratégia EMANUEL$_{FS}$, pois, antes da construção do modelo de classificação, passou-se a realizar a seleção de atributos. Essa dificuldade foi sentida principalmente em conjuntos de dados de alta dimensionalidade, que acabam por elevar o tempo de execução.

Outra limitação também relacionada ao tempo de execução foi a coleta dos metadados para a construção dos meta-conjuntos de dados. Como o hiperespaço multirrótulo contém

diferentes algoritmos, com espaço de hiperparâmetros amplo, a quantidade de combinações que geram algoritmos únicos é elevada. Dessa forma, foram necessárias repetidas execuções dos algoritmos EMANUEL e MORS para a coleta dos metadados utilizados na construção dos modelos substitutos de EMANUEL$_{SM}$. Com a inclusão do hiperespaço de seleção de atributos, a quantidade de possíveis soluções candidatas aumentou ainda mais. Isso exigiu um número maior de execuções de EMANUEL$_{FS}$ e MORS para a coleta dos metadados utilizados em EMANUEL$_{FS+}$.

Por fim, as limitações estiveram relacionadas aos algoritmos de seleção de atributos no cenário multirrótulo. Encontramos um número limitado de bibliotecas que disponibilizam esses algoritmos gratuitamente. `PyIT-MLFS` foi a única biblioteca com algoritmos de seleção de atributos multirrótulo. `Scikit-learn` possui vários algoritmos de seleção de atributos para o cenário monorrótulo. Alguns desses algoritmos, como o *SelectFromModel* e RFE, quando executados com algoritmos base específicos (como RF e DT, por exemplo), têm suporte à classificação multirrótulo. Retomando ao tempo de execução, alguns algoritmos de seleção de atributos multirrótulo têm tempo de execução elevado, tornando inviável a execução do AutoML sem limitar o tempo de execução desses algoritmos.

10.3 Trabalhos Futuros

A otimização multiobjetivo mostrou-se promissora ao encontrar soluções que otimizem simultaneamente mais de um objetivo. Como trabalhos futuros, sugerimos a exploração de novos objetivos de otimização, assim como a avaliação de novos métodos de seleção de soluções na fronteira de Pareto. Particularmente, a seleção de soluções localizadas na área central da fronteira é uma possibilidade de investigação, pois essas soluções tendem a apresentar bons resultados para os diferentes objetivos da otimização.

Os modelos substitutos demonstraram-se uma boa alternativa para o AutoML. No entanto, no Capítulo 9, conduzimos um experimento no formato prova de conceito para validar a hipótese de pesquisa, incluindo apenas cinco conjuntos de dados. Esse experimento poderia ser expandido, incorporando novos conjuntos de dados, o que permitiria a condução de testes estatíticos para comparação de EMANUEL$_{FS+}$ com os algoritmos *baselines*. Além disso, seria interessante desenvolver alternativas para os cenários nos quais novos conjuntos de dados são apresentados aos modelos substitutos. Uma possibilidade é que o novo conjunto de dados seja incluído em uma lista de futuros modelos substitutos a serem desenvolvidos e que o conhecimento acumulado em outras tarefas fosse utilizado, por meio de alguma técnica de meta-aprendizado, para contribuir com o AutoML.

O meta-aprendizado poderia ser empregado, por exemplo, para recomendar os algoritmos mais promissores do hiperespaço, guiando a busca por regiões mais promissoras. Essas técnicas são importantes principalmente quando pretende-se expandir a automatização das tarefas do *pipeline* do AM. Em relação a essas tarefas, tarefas relacionadas à

preparação dos dados e à engenharia de atributos ainda não foram totalmente automatizadas. Na tese, o pré-processamento dos conjuntos de dados foi realizado de forma manual, automatizamos a seleção de atributos, mas não abordamos a construção e a extração de atributos. Essas tarefas *pipeline* poderiam ser automatizadas. A estratégia EMANUEL$_{FS}$ poderia ser remodelada, melhorando os algoritmos de seleção e incluindo algoritmos de construção e extração de atributos.

Seria relevante realizar uma análise mais aprofundada dos resultados, investigando, por exemplo, a interpretabilidade dos modelos selecionados da fronteira de Pareto e o desempenho dos modelos de classificação em rótulos raros. Normalmente, EMANUEL$_{FS}$ induz modelos utilizando um número menor de atributos. Mas esses modelos são mais interpretáveis que os modelos encontrados por EMANUEL? Uma análise da interpretabilidade poderia responder a essa questão. Em problemas multirrótulo, o desbalanceamento por rótulo é um fator crítico, pois rótulos minoritários tendem a ser preditos com menor frequência, apesar de muitas vezes serem os mais relevantes para a tarefa. Essa investigação consistiria em verificar se as soluções encontradas por nossas estratégias AutoML conseguem capturar adequadamente os rótulos raros.

Um outro possível experimento consiste em investigar da influência relativa da abordagem multirrótulo e do algoritmo monorrótulo selecionados pela estratégia AutoML. Evidências empíricas reportadas por Rivolli et al. (2020) indicam que a escolha do algoritmo base tende a exercer maior impacto no desempenho final do classificador do que a abordagem multirrótulo adotada. A realização desse estudo permitiria quantificar a contribuição de cada componente e aprimorar o entendimento sobre os fatores que mais influenciam a qualidade dos modelos induzidos pelo AutoML, o que possibilitaria o refinamento dos hiperespaços utilizados. Por fim, todas as sugestões feitas nesta seção em relação ao AutoML para classificação multirrótulo poderiam ser replicadas para o AutoML em problemas de regressão multi-alvo.

Referências

ARAGÃO, M. V. C. et al. A practical evaluation of automl tools for binary, multiclass, and multilabel classification. **Scientific Reports**, v. 15, n. 1, p. 17682, May 2025. ISSN 2045-2322.

BABOR, M. et al. Application of non-dominated sorting genetic algorithm NSGA-II to increase the efficiency of bakery production: A case study. **Processes**, v. 10, n. 8, 2022. ISSN 2227-9717.

BLANK, J.; DEB, K. Pymoo: Multi-objective optimization in python. **IEEE Access**, v. 8, p. 89497–89509, 2020.

BOGATINOVSKI, J. et al. Comprehensive comparative study of multi-label classification methods. **Expert Systems with Applications**, v. 203, p. 117215, 2022. ISSN 0957-4174.

BOLÓN-CANEDO, V.; SÁNCHEZ-MAROÑO, N.; ALONSO-BETANZOS, A. A review of feature selection methods on synthetic data. **Knowledge and Information Systems**, v. 34, p. 483–519, mar 2012.

BRAZDIL, P. et al. **Metalearning: Applications to Automated Machine Learning and Data Mining**. [S.l.]: Springer International Publishing, 2022. (Cognitive Technologies). ISBN 9783030670238.

BROCHU, E.; CORA, V.; FREITAS, N. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. **CoRR**, abs/1012.2599, dec 2010.

CARNEIRO, D. et al. Using meta-learning to predict performance metrics in machine learning problems. **Expert Systems**, v. 40, n. 1, p. e12900, 2023.

CARVALHO, D. V.; PEREIRA, E. M.; CARDOSO, J. S. Machine learning interpretability: A survey on methods and metrics. **Electronics**, v. 8, n. 8, 2019. ISSN 2079-9292.

CHEN, Y.-W.; SONG, Q.; HU, X. Techniques for automated machine learning. **SIGKDD Explorations Newsletter**, Association for Computing Machinery, New York, NY, USA, v. 22, n. 2, p. 35–50, jan 2021. ISSN 1931-0145.

CHENG, W.; HÜLLERMEIER, E. Combining instance-based learning and logistic regression for multilabel classification. **Machine Learning**, Kluwer Academic Publishers, USA, v. 76, n. 2–3, p. 211—-225, sep 2009. ISSN 0885-6125.

CHERMAN, E. A. **Aprendizado de Máquina Multirrótulo: Explorando a Dependência de Rótulos e o Aprendizado Ativo**. Tese (Doutorado) — Instituto de Ciências Matemáticas e de Computação, 2014.

CHU, X. et al. Data cleaning: Overview and emerging challenges. In: **Proceedings of the 2016 International Conference on Management of Data**. New York, NY, USA: Association for Computing Machinery, 2016. (SIGMOD '16), p. 2201–2206. ISBN 9781450335317.

DANGETI, P. **Statistics for Machine Learning**. Birmingham, UK: Packt Publishing Ltd, 2017. ISBN 978-1-78398-575-1.

DEB, K.; DEB, K. Multi-objective optimization. In: _____. **Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques**. Boston, MA: Springer US, 2014. p. 403–449. ISBN 978-1-4614-6940-7.

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: Nsga-ii. **IEEE Transactions on Evolutionary Computation**, v. 6, n. 2, p. 182–197, 2002.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. **Journal of Machine Learning Research**, JMLR.org, v. 7, p. 1—30, dec 2006. ISSN 1532-4435.

DOQUIRE, G.; VERLEYSEN, M. Feature selection for multi-label classification problems. In: CABESTANY, J.; ROJAS, I.; JOYA, G. (Ed.). **Advances in Computational Intelligence**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 9–16. ISBN 978-3-642-21501-8.

EGGENSPERGER, K. et al. Efficient benchmarking of algorithm configurators via model-based surrogates. **Machine Learning**, Kluwer Academic Publishers, USA, v. 107, n. 1, p. 15–41, jan. 2018. ISSN 0885-6125.

ELSKEN, T. et al. Neural architecture search: A survey. **Journal of Machine Learning Research**, v. 20, n. 55, p. 1–21, 2019.

ESKANDARI, S. Pyit-mlfs: a python-based information theoretical multi-label feature selection library. **International Journal of Research in Industrial Engineering**, Ayandegan Institute of Higher Education, v. 11, n. 1, p. 9–15, 2022. ISSN 2783-1337.

EVCHENKO, M. **Frugal Learning: Applying Machine Learning with Minimal Resources**. Tese (Doutorado) — Eindhoven University of Technology, Eindhoven, Holanda, 2016. PhD Thesis.

FANG, Y.; LI, J. A review of tournament selection in genetic programming. In: CAI, Z. et al. (Ed.). **Advances in Computation and Intelligence**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 181–192. ISBN 978-3-642-16493-4.

FENG, J.; SHEN, W. Z.; XU, C. Multi-objective random search algorithm for simultaneously optimizing wind farm layout and number of turbines. **Journal of Physics: Conference Series**, v. 753, sep 2016.

FERNÁNDEZ-DELGADO, M. et al. Do we need hundreds of classifiers to solve real world classification problems? **Journal of Machine Learning Research**, JMLR.org, v. 15, n. 1, p. 3133–3181, jan 2014. ISSN 1532-4435.

FEURER, M.; HUTTER, F. Hyperparameter optimization. In: _____. **Automated Machine Learning: Methods, Systems, Challenges**. Cham: Springer International Publishing, 2019. p. 3–33. ISBN 978-3-030-05318-5.

FEURER, M. et al. Efficient and robust automated machine learning. In: **Advances in Neural Information Processing Systems 28**. [S.l.: s.n.], 2015. p. 2962–2970.

FEURER, M.; SPRINGENBERG, J. T.; HUTTER, F. Initializing bayesian hyperparameter optimization via meta-learning. In: **Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence**. [S.l.]: AAAI Press, 2015. (AAAI'15), p. 1128–1135. ISBN 0262511290.

FONSECA, C. M.; PAQUETE, L.; LÓPEZ-IBÁNEZ, M. An improved dimension-sweep algorithm for the hypervolume indicator. In: IEEE. **2006 IEEE international conference on evolutionary computation**. [S.l.], 2006. p. 1157–1163.

GARCIA, L. P. et al. Boosting meta-learning with simulated data complexity measures. **Intelligent Data Analysis**, IOS Press, NLD, v. 24, n. 5, p. 1011–1028, jan 2020. ISSN 1088-467X.

GARCÍA-PEDRAJAS, N. E. et al. A thorough experimental comparison of multilabel methods for classification performance. **Pattern Recognition**, v. 151, p. 110342, 2024. ISSN 0031-3203.

GEORGIEVSKI, I.; AIELLO, M. Htn planning: Overview, comparison, and beyond. **Artificial Intelligence**, v. 222, p. 124–156, 2015. ISSN 0004-3702.

GIJSBERS, P.; VANSCHOREN, J. Gama: A general automated machine learning assistant. In: DONG, Y. et al. (Ed.). **Machine Learning and Knowledge Discovery in Databases. Applied Data Science and Demo Track**. Cham: Springer International Publishing, 2021. p. 560–564. ISBN 978-3-030-67670-4.

GOMES, T. A. et al. Combining meta-learning and search techniques to select parameters for support vector machines. **Neurocomputing**, v. 75, n. 1, p. 3–13, 2012. ISSN 0925-2312. Brazilian Symposium on Neural Networks (SBRN 2010) International Conference on Hybrid Artificial Intelligence Systems (HAIS 2010).

GUO, Y.; GU, S. Multi-label classification using conditional dependency networks. In: **Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two**. [S.l.]: AAAI Press, 2011. (IJCAI'11), p. 1300–1305. ISBN 978-1-57735-514-4.

HALL, M. et al. The weka data mining software: an update. **SIGKDD Explorations Newsletter**, New York, NY, USA, v. 11, n. 1, p. 10–18, 2009. ISSN 1931-0145.

HASAN, R. et al. Towards transparent diabetes prediction: Combining automl and explainable ai for improved clinical insights. **Information**, v. 16, n. 1, 2025. ISSN 2078-2489.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction**. [S.l.]: Springer, 2009. (Springer series in statistics). ISBN 9780387848846.

HE, X.; ZHAO, K.; CHU, X. Automl: A survey of the state-of-the-art. **Knowledge-Based Systems**, v. 212, p. 106622, 2021. ISSN 0950-7051.

HEAD, T. et al. **holgern/scikit-optimize**. [S.l.]: Zenodo, 2024.

HOOF, J. V.; VANSCHOREN, J. Hyperboost: Hyperparameter optimization by gradient boosting surrogate models. **arXiv preprint**, abs/2101.02289, 2021.

KARL, F. et al. Multi-objective hyperparameter optimization in machine learning—an overview. **ACM Transactions on Evolutionary Learning and Optimization**, Association for Computing Machinery, v. 3, n. 4, 2023.

KARMAKER, S. K. et al. Automl to date and beyond: Challenges and opportunities. **ACM Computing Surveys**, Association for Computing Machinery, New York, NY, USA, v. 54, n. 8, oct 2021. ISSN 0360-0300.

KASHEF, S.; NEZAMABADI-POUR, H.; NIKPOUR, B. Multilabel feature selection: A comprehensive review and guiding experiments. **WIREs Data Mining and Knowledge Discovery**, v. 8, n. 2, p. e1240, 2018.

LE, T. T.; FU, W.; MOORE, J. H. Scaling tree-based automated machine learning to biomedical big data with a feature set selector. **Bioinformatics**, Oxford University Press, v. 36, n. 1, p. 250–256, 2020.

LEE, J.; KIM, D.-W. Scls: Multi-label feature selection based on scalable criterion for large label set. **Pattern Recognition**, v. 66, p. 342–352, 2017. ISSN 0031-3203.

LENT, M. van; FISHER, W.; MANCUSO, M. An explainable artificial intelligence system for small-unit tactical behavior. In: **Proceedings of the 16th Conference on Innovative Applications of Artifical Intelligence**. [S.l.]: AAAI Press, 2004. (IAAI'04), p. 900–907. ISBN 0262511835.

LI, L. et al. Multi-label feature selection via information gain. In: LUO, X.; YU, J. X.; LI, Z. (Ed.). **Advanced Data Mining and Applications**. Cham: Springer International Publishing, 2014. p. 345–355. ISBN 978-3-319-14717-8.

LI, M.; YAO, X. Quality evaluation of solution sets in multiobjective optimisation: A survey. **ACM Computing Surveys**, Association for Computing Machinery, New York, NY, USA, v. 52, n. 2, mar 2019. ISSN 0360-0300.

LIAN, D. et al. Towards fast adaptation of neural architectures with meta learning. In: **International Conference on Learning Representations**. [S.l.: s.n.], 2019.

LIU, H. et al. Hierarchical representations for efficient architecture search. In: **International Conference on Learning Representations**. [S.l.: s.n.], 2018.

LORENA, A. C. et al. Data complexity meta-features for regression problems. **Machine Learning**, Kluwer Academic Publishers, USA, v. 107, n. 1, p. 209–246, jan. 2018. ISSN 0885-6125.

MADJAROV, G. et al. An extensive experimental comparison of methods for multi-label learning. **Pattern Recognition**, v. 45, n. 9, p. 3084–3104, 2012. ISSN 0031-3203. Best Papers of Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA'2011).

MANTOVANI, R. G. **Use of meta-learning for hyperparameter tuning of classification problems**. Tese (Doutorado) — Universidade de São Paulo, São Carlos, 2018. Orientador: André Carlos Ponce de Leon Ferreira de Carvalho; Coorientador: Joaquin Vanschoren.

MANTOVANI, R. G. et al. A meta-learning recommender system for hyperparameter tuning: Predicting when tuning improves svm classifiers. **Information Sciences**, v. 501, p. 193–221, 2019. ISSN 0020-0255.

MCKAY, R. I. et al. Grammar-based genetic programming: a survey. **Genetic Programming and Evolvable Machines**, Kluwer Academic Publishers, USA, v. 11, n. 3–4, p. 365–396, set. 2010. ISSN 1389-2576.

MCKINNEY Wes. Data Structures for Statistical Computing in Python. In: WALT Stéfan van der; MILLMAN Jarrod (Ed.). **Proceedings of the 9th Python in Science Conference**. [S.l.: s.n.], 2010. p. 56 – 61.

MILES, J. R-squared, adjusted r-squared. In: **Encyclopedia of Statistics in Behavioral Science**. [S.l.]: John Wiley & Sons, 2005. ISBN 978-1-119-45078-8.

MILLER, T. Explanation in artificial intelligence: Insights from the social sciences. **Artificial Intelligence**, v. 267, p. 1–38, 2019. ISSN 0004-3702.

MITCHELL, T. M. **Machine Learning**. New York: McGraw Hill, 1997.

MOHR, F.; WEVER, M.; HüLLERMEIER, E. ML-Plan: Automated machine learning via hierarchical planning. **Machine Learning**, Kluwer Academic Publishers, USA, v. 107, n. 8–10, p. 1495–1515, set. 2018. ISSN 0885-6125.

MOLNAR, C. **Interpretable Machine Learning**: A guide for making black box models explainable. 2. ed. [S.l.: s.n.], 2022.

MOYANO, J. M. et al. Review of ensembles of multi-label classifiers: Models, experimental study and prospects. **Information Fusion**, v. 44, p. 33–45, 2018. ISSN 1566-2535.

MURDOCH, W. J. et al. Definitions, methods, and applications in interpretable machine learning. **Proceedings of the National Academy of Sciences**, v. 116, n. 44, p. 22071–22080, 2019.

PAN, S. J.; YANG, Q. A survey on transfer learning. **IEEE Transactions on Knowledge and Data Engineering**, v. 22, n. 10, p. 1345–1359, 2010.

PAULLADA, A. et al. Data and its (dis)contents: A survey of dataset development and use in machine learning research. **Patterns**, v. 2, n. 11, p. 100336, 2021. ISSN 2666-3899.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, JMLR.org, v. 12, p. 2825–2830, 2011.

PEREIRA, R. B. et al. Correlation analysis of performance measures for multi-label classification. **Information Processing and Management**, v. 54, n. 3, p. 359–369, 2018. ISSN 0306-4573.

READ, J. et al. Classifier chains for multi-label classification. In: BUNTINE, W. et al. (Ed.). **Machine Learning and Knowledge Discovery in Databases**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 254–269. ISBN 978-3-642-04174-7.

_____. MEKA: A multi-label/multi-target extension to Weka. **Journal of Machine Learning Research**, v. 17, n. 21, p. 1–5, 2016.

REIF, M. et al. Automatic classifier selection for non-experts. **Pattern Analysis and Applications**, Springer-Verlag, Berlin, Heidelberg, v. 17, n. 1, p. 83–96, fev. 2014. ISSN 1433-7541.

RIBEIRO, M. T.; SINGH, S.; GUESTRIN, C. "why should i trust you?": Explaining the predictions of any classifier. In: **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: Association for Computing Machinery, 2016. (KDD '16), p. 1135–1144. ISBN 9781450342322.

RIDD, P.; GIRAUD-CARRIER, C. Using metalearning to predict when parameter optimization is likely to improve classification accuracy. In: **Proceedings of the 2014 International Conference on Meta-Learning and Algorithm Selection - Volume 1201**. Aachen, DEU: CEUR-WS.org, 2014. (MLAS'14), p. 18–23. ISBN 16130073.

RIVOLLI, A. et al. An empirical analysis of binary transformation strategies and base algorithms for multi-label learning. **Mach. Learn.**, Kluwer Academic Publishers, USA, v. 109, n. 8, p. 1509–1563, ago. 2020. ISSN 0885-6125.

ROH, Y.; HEO, G.; WHANG, S. E. A survey on data collection for machine learning: A big data - ai integration perspective. **IEEE Transactions on Knowledge and Data Engineering**, v. 33, n. 4, p. 1328–1347, 2021.

SÁ, A. G. C. de; FREITAS, A. A.; PAPPA, G. L. Automated selection and configuration of multi-label classification algorithms with grammar-based genetic programming. In: AUGER, A. et al. (Ed.). **Parallel Problem Solving from Nature — PPSN XV**. [S.l.]: Springer, 2018, (Lecture Notes in Computer Science). p. 308–320.

SÁ, A. G. C. de; PAPPA, G. L.; FREITAS, A. A. Towards a method for automatically selecting and configuring multi-label classification algorithms. In: **Proceedings of the Genetic and Evolutionary Computation Conference Companion**. New York, NY, USA: Association for Computing Machinery, 2017. (GECCO '17), p. 1125–1132. ISBN 9781450349390.

SÁ, A. G. C. de et al. A robust experimental evaluation of automated multi-label classification methods. **Proceedings of the 2020 Genetic and Evolutionary Computation Conference**, ACM, jun 2020.

SANTAFE, G.; INZA, I. n.; LOZANO, J. A. Dealing with the evaluation of supervised classification algorithms. **Artificial Intelligence Review**, Kluwer Academic Publishers, USA, v. 44, n. 4, p. 467–508, dez. 2015. ISSN 0269-2821.

SCHNEIDER, L.; BISCHL, B.; THOMAS, J. Multi-objective optimization of performance and interpretability of tabular supervised machine learning models. In: **Proceedings of the Genetic and Evolutionary Computation Conference**. New York, NY, USA: Association for Computing Machinery, 2023. (GECCO '23), p. 538–547. ISBN 9798400701191.

SECHIDIS, K.; TSOUMAKAS, G.; VLAHAVAS, I. On the stratification of multi-label data. In: GUNOPULOS, D. et al. (Ed.). **Machine Learning and Knowledge Discovery in Databases**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 145–158. ISBN 978-3-642-23808-6.

SHARMA, s.; CHAHAR, V. A comprehensive review on multi-objective optimization techniques: Past, present and future. **Archives of Computational Methods in Engineering**, v. 29, p. 3, jul 2022.

SPOLAÔR, N. et al. A comparison of multi-label feature selection methods using the problem transformation approach. **Electronic Notes in Theoretical Computer Science**, v. 292, p. 135–151, 2013. ISSN 1571-0661. Proceedings of the XXXVIII Latin American Conference in Informatics (CLEI).

SZYMANSKI, P.; KAJDANOWICZ, T. Scikit-multilearn: a scikit-based python environment for performing multi-label classification. **Journal of Machine Learning Research**, JMLR.org, v. 20, n. 1, p. 209–230, jan 2019. ISSN 1532–4435.

TSOUMAKAS, G.; KATAKIS, I.; VLAHAVAS, I. Mining multi-label data. In: **Data mining and knowledge discovery handbook**. [S.l.]: Springer, 2009. p. 667–685.

_____. Random k-labelsets for multilabel classification. **IEEE Transactions on Knowledge and Data Engineering**, IEEE Educational Activities Department, USA, v. 23, n. 7, p. 1079–1089, jul 2011. ISSN 1041-4347.

TSOUMAKAS, G. et al. Mulan: A java library for multi-label learning. **Journal of Machine Learning Research**, v. 12, p. 2411–2414, 2011.

TSOUMAKAS, G.; VLAHAVAS, I. Random k-labelsets: An ensemble method for multilabel classification. In: KOK, J. N. et al. (Ed.). **Machine Learning: ECML 2007**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 406–417. ISBN 978-3-540-74958-5.

URBANOWICZ, R. J. et al. Benchmarking relief-based feature selection methods for bioinformatics data mining. **Journal of Biomedical Informatics**, v. 85, p. 168–188, 2018. ISSN 1532-0464.

VALLE, A. M. D.; MANTOVANI, R. G.; CERRI, R. Autommlc: An automated and multi-objective method for multi-label classification. In: **12th Brazilian Conference Intelligent Systems, BRACIS 2023, Part II**. [S.l.: s.n.], 2023. p. 291–306. ISBN 978-3-031-45388-5.

_____. A systematic literature review on automl for multi-target learning tasks. **Artif. Intell. Rev.**, Kluwer Academic Publishers, USA, v. 56, n. Suppl 2, p. 2013–2052, ago. 2023. ISSN 0269-2821. Disponível em: <https://doi.org/10.1007/s10462-023-10569-2>.

VANSCHOREN, J. Meta-learning. In: _____. **Automated Machine Learning: Methods, Systems, Challenges**. Cham: Springer International Publishing, 2019. p. 35–61. ISBN 978-3-030-05318-5.

VERMA, S.; PANT, M.; SNASEL, V. A comprehensive review on NSGA-II for multi-objective combinatorial optimization problems. **IEEE Access**, v. 9, p. 57757–57791, 2021.

WANG, Z. et al. Principles, research status, and prospects of feature engineering for data-driven building energy prediction: A comprehensive review. **Journal of Building Engineering**, v. 58, p. 105028, 2022. ISSN 2352-7102.

WEBB, A. R. A. R. **Statistical pattern recognition**. 3rd ed. / andrew r. webb, keith d. copsey, gavin cawley.. ed. Oxford: Wiley-Blackwell, 2011. ISBN 9781119952954.

WEN, Y.-W.; PENG, S.-H.; TING, C.-K. Two-stage evolutionary neural architecture search for transfer learning. **IEEE Transactions on Evolutionary Computation**, IEEE, v. 25, n. 5, p. 928–940, 2021.

WEVER, M. et al. Automl for multi-label classification: Overview and empirical evaluation. **IEEE Transactions on Pattern Analysis & Machine Intelligence**, IEEE Computer Society, Los Alamitos, CA, USA, v. 43, n. 09, p. 3037–3054, sep 2021. ISSN 1939-3539.

WEVER, M. D.; MOHR, F.; HÜLLERMEIER, E. Ml-plan for unlimited-length machine learning pipelines. In: **ICML 2018 AutoML Workshop**. [S.l.: s.n.], 2018.

WEVER, M. D. et al. Automating multi-label classification extending ML-Plan. In: **Proceedings of the 6th ICML Workshop on Automated Machine Learning**. [S.l.: s.n.], 2019. (ICML '19).

WIECZORKOWSKA, A.; SYNAK, P.; RAŚ, Z. W. Multi-label classification of emotions in music. In: KŁOPOTEK, M. A.; WIERZCHOŃ, S. T.; TROJANOWSKI, K. (Ed.). **Intelligent Information Processing and Web Mining**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 307–315. ISBN 978-3-540-33521-4.

XU, D. et al. Survey on multi-output learning. **IEEE Transactions on Neural Networks and Learning Systems**, v. 31, n. 7, p. 2409–2429, 2020.

YAO, Q. et al. Taking human out of learning applications: A survey on automated machine learning. **CoRR**, abs/1810.13306, 2018.

ZHANG, M.-L.; ZHOU, Z.-H. A k-nearest neighbor based algorithm for multi-label classification. In: **2005 IEEE International Conference on Granular Computing**. [S.l.: s.n.], 2005. v. 2, p. 718–721.

_____. Multilabel neural networks with applications to functional genomics and text categorization. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 18, n. 10, p. 1338–1351, 2006.

_____. A review on multi-label learning algorithms. **IEEE Transactions on Knowledge and Data Engineering**, v. 26, n. 8, p. 1819–1837, 2014.

ZHANG, P.; LIU, G.; GAO, W. Distinguishing two types of labels for multi-label feature selection. **Pattern Recognition**, v. 95, p. 72–82, 2019. ISSN 0031-3203.

ZHANG, P. et al. Multi-label feature selection considering label supplementation. **Pattern Recognition**, v. 120, p. 108137, 2021. ISSN 0031-3203.

ZHONGGUO, Y. et al. Choosing classification algorithms and its optimum parameters based on data set characteristics. **Journal of Computers**, v. 28, n. 5, p. 26–38, 2017.

ZHUANG, F. et al. A comprehensive survey on transfer learning. **Proceedings of the IEEE**, IEEE, v. 109, n. 1, p. 43–76, 2020.

ZITZLER, E.; DEB, K.; THIELE, L. Comparison of multiobjective evolutionary algorithms: Empirical results. **Evolutionary Computation**, MIT Press, Cambridge, MA, USA, v. 8, n. 2, p. 173–195, jun 2000. ISSN 1063-6560.

ZITZLER, E.; THIELE, L. Multiobjective optimization using evolutionary algorithms — a comparative case study. In: EIBEN, A. E. et al. (Ed.). **Parallel Problem Solving from Nature — PPSN V**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. p. 292–301. ISBN 978-3-540-49672-4.

ZÖLLER, M.; HUBER, M. F. Benchmark and survey of automated machine learning frameworks. **Journal of Artificial Intelligence Research**, AI Access Foundation, El Segundo, CA, USA, v. 70, p. 409—472, maio 2021. ISSN 1076-9757.

Apêndices

APÊNDICE A

Hiperespaço para Classificação Multirrótulo

O hiperespaço foi definido com base no hiperespaço de tamanho médio proposto por Sá et al. (2020). As Seções A.1 e A.2 apresentam os hiperespaços monorrótulo e multirrótulo, respectivamente. Para mais informações sobre os algoritmos do hiperespaço, consulte Sá et al. (2020).

A.1 Hiperespaço Monorrótulo

O hiperespaço monorrótulo contém 18 algoritmos de classificação monorrótulo da biblioteca `WEKA` (HALL et al., 2009), conforme lista a Tabela 18. Os hiperparâmetros são representados por meio de vetores. Para espaços de hiperparâmetros numéricos, informamos o intervalo de valores que um hiperparâmetro pode assumir. Quando o incremento do intervalo não é informado, consideramos o incremento de 1. Um dos hiperparâmetros do classificador monorrótulo SMO é o *kernel*, os possíveis valores desse hiperparâmetro estão na Tabela 17. A normalização pode ocorrer para algoritmos baseados em distância, como os algoritmos IBk, SMO, K*, MLP e VP.

Tabela 17 – Hiperespaço de *kernels* do algoritmo SMO.

#	**Kernel**	**Hiperparâmetros**
1	Normalized PolyKernel	Exponent (exp)[-E]: $\{exp \in \mathbb{R} \mid 0.2 \leq exp \leq 5.0 \text{ step } 0.1\}$ Use Lower-Order (ulo)[-L]: $ulo \in \{False, True\}$

continua na próxima página

Tabela 17 – continuação

#	Kernel	Hiperparâmetros
2	PolyKernel	Exponent (exp)[-E]: $\{exp \in \mathbb{R} \mid 0.2 \leq exp \leq 5.0 \text{ step } 0.1\}$ Use Lower-Order (ulo)[-L]: $ulo \in \{False, True\}$
3	Puk	Omega (om)[-O]: $\{om \in \mathbb{R} \mid 0.1 \leq om \leq 1.0 \text{ step } 0.1\}$ Sigma (sig)[-S]: $\{sig \in \mathbb{R} \mid 0.1 \leq sig \leq 10.0 \text{ step } 0.1\}$
4	RBFKernel	Gamma (g)[-G]: $g \in \mathbb{R} \mid 0.0001 \leq g \leq 1.0 \text{ step } *10$

Fonte: Sá et al. (2020)

Tabela 18 – Hiperespaço monorrótulo. Na tabela, FEATURES é o número de atributos e LABELS o número de rótulos do conjunto de dados multirrótulo.

#	Algoritmo	Hiperparâmetros
1	BNC	Normalize (norm): $norm \in \{False\}$ Not use ADTree (adt)[-D]: $adt \in \{False\}$ Search Method (sm)[-Q]:$sm \in$ {Tree Augmented Naïve Bayes (TAN), K2, Hill Climbing (HC), Look Ahead in Good Directions Hill Climbing (LAGDHC), Tabu Search (TS)}
2	J48	Normalize (norm): $norm \in \{False\}$ Minimum number of objects (mno)[-M]: $\{mno \in \mathbb{Z} \mid 1 \leq mno \leq 64\}$ Binary splits (bs)[-B]: $bs \in \{False, True\}$ Use MDL correction (umc)[-J]: $umc \in \{False, True\}$ Use Laplace (ul)[-A]: $ul \in \{False, True\}$ Unpruned (u)[-U]: $u \in \{False, True\}$ Confidence factor (cf)[-C]: $\{cf \in \mathbb{R} \mid 0.0 \leq cf \leq 1.0 \text{ step } 0.05\}$ $if - U == False$ Collapse tree (ct)[-O]: $ct \in \{False, True\}$ $if - U == False$ Subtree raising (sr)[-S]: $sr \in \{False, True\}$ $if - U == False$
3	DT	Normalize (norm): $norm \in \{False\}$ Evaluation Measure (em)[-E]: $em \in$ {accuracy (acc), root mean squared error (rmse), mean absolute error (mae), area under the ROC curve (auc)} Use IBk (uibk)[-I]: $uibk \in \{False, True\}$ Search method (sm)[-S]: $sm \in$ {'BestFirst', 'GreedyStepwise'} Cross-Validation (crv)[-X]: $crv \in \{1, 2, 3, 4\}$
4	IBk	Normalize (norm): $norm \in \{False, True\}$ Number of neighbors (k)[-K]: $\{k \in \mathbb{Z} \mid 1 \leq k \leq 64\}$ Leave-one-out (loo)[-X]: $loo \in \{False, True\}$ Weight neighbours by the inverse of their distance (wi) [-I]: $wi \in \{False, True\}$ Weight neighbours by 1 - their distance (wd)[-F]: $wd \in \{False, True\}$ $if - I == False$
5	JRip	Normalize (norm): $norm \in \{False, True\}$ Minimum total weight (mtw)[-N]: $\{mtw \in \mathbb{R} \mid 1.0 \leq mtw \leq 5.0 \text{ step } 0.1\}$ Check error rate (cer)[-E]: $cer \in \{False, True\}$ Use pruning (up)[-P]: $up \in \{False, True\}$ Optimizations (o)[-O]: $\{o \in \mathbb{Z} \mid 1 \leq o \leq 5\}$
6	K*	Normalize (norm): $norm \in \{False, True\}$ Global blending (gb)[-B]: $\{gb \in \mathbb{Z} \mid 1 \leq o \leq 100\}$

continua na próxima página

#	Algoritmo	Hiperparâmetros
		Entropic auto-blending (eab)[-E]: $eab \in \{False\}$ Missing Mode (mm)[-M]: $mm \in$ {average column entropy curves (a), ignore the instances with missing values (d), treat missing values asmaximally different (m), normalize over the attributes (n)}
7	LMT	Normalize (norm): $norm \in \{False\}$ Minimum number of objects (mno)[-M]: $\{mno \in \mathbb{Z} \vert 1 \leq mno \leq 64\}$ Convert Nominal (cn)[-B] $cn \in \{False, True\}$ Split on residuals (sor)[-R]: $sor \in \{False, True\}$ Fast Regression (fr)[-C]: $fr \in \{False, True\}$ Error on probabilities (eop)[-P]: $eop \in \{False, True\}$ Use AIC (uaic)[-A]: $uaic \in \{False, True\}$ Weight trim beta(wtb)[-W]: $\{wtb \in \mathbb{R} \vert 0.0 \leq wtb \leq 1.0 \text{ step } 0.01\}$
8	LR	Normalize (norm): $norm \in \{False\}$ Ridge (r)[-R]: $\{r \in \mathbb{R} \vert 10^{-12} \leq r \leq 10 \text{ step } *10\}$
9	MLP	Normalize (norm): $norm \in \{False, True\}$ Learning rate (lrt)[-L]: $\{lrt \in \mathbb{R} \vert 0.1 \leq lrt \leq 1.0 \text{ step } 0.1\}$ Momentum (m)[-M]: $\{m \in \mathbb{R} \vert 0 \leq m \leq 1.0 \text{ step } 0.1\}$ Number of hidden nodes (nhn)[-H]: $nhn \in \{(FEATURES + LABELS)/2, FEATURES, LABELS, FEATURES + LABELS\}$ Nominal to binary filter (n2b)[-B]: $n2b \in \{False, True\}$ Reset (r)[-R]: $r \in \{False, True\}$ Decay (d)[-D]: $d \in \{False, True\}$
10	NB	Normalize (norm): $norm \in \{False\}$ Use kernel estimator (uke)[-K]: $uke \in \{False, True\}$ Use supervised distribution (usd)[-D]: $usd \in \{False, True\}\ if - K == False$
11	PART	Normalize (norm): $norm \in \{False\}$ Minimum number of objects (mno)[-M]: $\{mno \in \mathbb{Z} \vert 1 \leq mno \leq 64\}$ Binary splits (bs)[-B]: $bs \in \{False, True\}$ Reduced-error pruning (rep)[-R]: $rep \in \{False, True\}$ Number of folds (nr)[-N]: $nr \in \{2, 3, 4, 5\}\ if - R == True$
12	REPTree	Normalize (norm): $norm \in \{False\}$ Minimum weight (mw)[-M]: $\{mw \in \mathbb{Z} \vert 1 \leq mno \leq 64\}$ Maximum depth (md)[-L]: 50% of the probability of md is -1, and 50% of the probability of nd is a number in $\{md \in \mathbb{Z} \vert 2 \leq nf \leq 20\}$ Use pruning (up)[-P]: $up \in \{False, True\}$
13	RF	Normalize (norm): $norm \in \{False\}$ Number of trees (nt)[-I]: $\{nt \in \mathbb{Z} \vert 2 \leq nt \leq 256\}$ Number of features (nf)[-K]: 50% of the probability of nf is 0, and 50% of the probability of nf is a number in $\{nf \in \mathbb{Z} \vert 2 \leq nf \leq 32\}$ Maximum depth (md)[-depth]: 50% of the probability of md is 0, and 50% of the probability of md is a number in $\{md \in \mathbb{Z} \vert 2 \leq md \leq 20\}$
14	RT	Normalize (norm): $norm \in \{False\}$ Minimum weight (mw)[-M]: $\{mw \in \mathbb{Z} \vert 1 \leq mno \leq 64\}$ Number of features (nf)[-K]: 50% of the probability of nf is 0, and 50% of the probability of nf is a number in $\{nf \in \mathbb{Z} \vert 2 \leq nf \leq 32\}$ Maximum depth (md)[-depth]: 50% of the probability of md is 0, and 50% of the probability of md is a number in $\{md \in \mathbb{Z} \vert 2 \leq md \leq 20\}$ Number of folds for back-fitting and for growing the tree (nfbgt)[-N]: 50% of the probability of nfbgt is 0, and 50% of the probability of $nfbgt \in \{2, 3, 4, 5\}$

continua na próxima página

#	Algoritmo	Hiperparâmetros
15	SGD	Normalize (norm): $norm \in \{False\}$ Loss function (lf)[-F]: $lf \in \{0, 1\}$ Learning rate (lrt)[-L]: $\{lrt \in \mathbb{R} \mid 1.e-05 \leq mno \leq 1.e-01 \text{ step } *10\}$ Ridge (r)[-R]: $\{r \in \mathbb{R} \mid 1.e-12 \leq r \leq 10 \text{ step } *10\}$ Do not normalize (nn)[-N]: $nn \in \{False, True\}$ Do not replace missing values (nrmv)[-M]: $nrmv \in \{False, True\}$
16	SL	Normalize (norm): $norm \in \{False\}$ Weight trim beta (wtb)[-W]: $\{wtb \in \mathbb{R} \mid 0.0 \leq wtb \leq 1.0 \text{ step } 0.1\}$ Use Cross-Validation (ucv)[-S]: $ucv \in \{False, True\}$ Use AIC (uaic)[-A]: $uaic \in \{False, True\}$
17	SMO	Normalize (norm): $norm \in \{False, True\}$ Cost (c)[-C]: $\{c \in \mathbb{R} \mid 0.5 \leq c \leq 1.5 \text{ step } 0.1\}$ Filter type (ft)[-N]: $ft \in \{0, 1, 2\}$ Build Calibration Models (bcm)[-M]: $bcm \in \{False, True\}$ Kernel(k)[-K]: one of the kernels in the hyperspace of SMO algorithm kernels
18	VP	Normalize (norm): $norm \in \{False, True\}$ Number of iterations (i)[-I]: $\{i \in \mathbb{Z} \mid 1 \leq i \leq 10\}$ Max K(mk)[-M]: $\{mk \in \mathbb{Z} \mid 5000 \leq mk \leq 50000\}$ Exponent (e)[-E]: $\{e \in \mathbb{R} \mid 0.2 \leq e \leq 5.0 \text{ step } 0.1\}$

Fonte: Sá et al. (2020)

A.2 Hiperespaço Multirrótulo

Inicialmente, o hiperespaço multirrótulo continha 18 algoritmos de classificação multirrótulo da biblioteca `MEKA` (READ et al., 2016). Posteriormente, o hiperespaço foi estendido, incluindo mais 3 algoritmos da biblioteca `MULAN` (TSOUMAKAS et al., 2011). As Tabelas 19 e 20 listam os algoritmos do hiperespaço multirrótulo.

Assim como no hiperespaço monorrótulo, os hiperparâmetros do hiperespaço multirrótulo foram representados por meio de vetores com valores fixos. O hiperespaço inclui os algoritmos do hiperespaço monorrótulo (Tabela 18) como classificadores base e os valores da função `Payoff` como valores de hiperparâmetros. Os classificadores base dos algoritmos FW, LP, PS, PSt, RAkEL, RAkELd e RT devem ser multiclasse. A normalização pode ocorrer para algoritmos baseados em distância, como os algoritmos BPNN e MLkNN.

```
payoff_function = {
    Accuracy, Jaccard index, Hamming score, Exact match, Jaccard distance,
    Hamming loss, ZeroOne loss, Harmonic score, One error, Rank loss,
    Avg precision, Log Loss (lim. L), Log Loss (lim. D), Micro Precision,
    Micro Recall, Macro Precision, Macro Recall, F1 (micro averaged),
    F1 (macro averaged by example), F1 (macro averaged by label),
    AUPRC (macro averaged), AUROC (macro averaged), Levenshtein distance
}
```

Tabela 19 – Hiperespaço multirrótulo. Na tabela, FEATURES é o número de atributos e LABELS o número de rótulos do conjunto de dados multirrótulo.

#	**Algoritmo**	**Hiperparâmetros**
1	BPNN	Normalize data (norm): $norm \in \{False, True\}$ Number of epochs (ne) [-E]: $\{ne \in \mathbb{Z} \mid 10 \leq ne \leq 1000\}$ Number of hidden units (nhu) [-H]: $\{nhu \in \mathbb{Z} \mid 0.2 * FEATURES \leq nhu \leq FEATURES\}$ Learning rate (lrt) [-r]: $\{lr \in \mathbb{R} \mid 0.001 \leq lr \leq 0.1 \text{ step } 0.001\}$ Momentum (m) [-m]: $\{m \in \mathbb{R} \mid 0.1 \leq m \leq 0.8 \text{ step } 0.1\}$
2	BCC	Base classifier (bc)[-W]: any classifier of single-label hyperspace Dependency type (dp)[-X]: $dp \in \{$C, I, Ib, Ibf, H, Hbf, X, F, L, None$\}$
3	BR	Base classifier (bc)[-W]: any classifier of single-label hyperspace
4	BRq	Down-sample ratio (dsr)[-P]: $\{dsr \in \mathbb{R} \mid 0.1 \leq dsr \leq 0.8 \text{ step } 0.005\}$ Base classifier (bc)[-W]: any classifier of single-label hyperspace
5	CC	Base classifier (bc)[-W]: any classifier of single-label hyperspace
6	CDN	Collection iterations (ci)[-Ic]: $\{ci \in \mathbb{Z} \mid 1 \leq ci \leq 100\}$ Iterations (i)[-I]: $\{i \in \mathbb{Z} \mid 100 \leq ci \leq 1000\}$ Base classifier (bc)[-W]: any classifier of single-label hyperspace
7	CDT	Width (w)[-H]: $w \in \{-1, 0, 1\}$ Dependency type (dp)[-X]: $dp \in \{$C, I, Ib, Ibf, H, Hbf, X, F, L, None$\}$ Iterations (i)[-I]: $\{i \in \mathbb{Z} \mid 1 < i \leq 100\}$ Collection iterations (ci)[-Is]: $\{ci \in \mathbb{Z} \mid 1 < ci \leq 100\}$ Density (d)[-L]: $\{d \in \mathbb{Z} \mid 1 \leq d \leq \sqrt{LABELS} + 1\}\ if - H == -1$ Base classifier (bc)[-W]: any classifier of single-label hyperspace
8	CT	Dependency type (dp)[-X]: $dp \in \{$C, I, Ib, Ibf, H, Hbf, X, F, L, None$\}$ Inference Iterations (ii)[-Iy]: $\{ii \in \mathbb{Z} \mid 1 < ii \leq 100\}$ Chain Iterations (chi)[-Is]: 50% of the probability of chi is 0, and 50% of the probability of chi is a number in $\{chi \in \mathbb{Z} \mid 1 < chi \leq 1500\}$ Payoff function (pof)[-P]): $pof \in$ payoff_function Width (w)[-H]: $w \in \{-1, 0, 1\}$ Density (d)[-L]: $\{d \in \mathbb{Z} \mid 1 \leq d \leq \sqrt{LABELS} + 1\}\ if - H == -1$ Base classifier (bc)[-W]: any classifier of single-label hyperspace
9	FW	Base classifier (bc)[-W]: any classifier of single-label hyperspace
10	LP	Base classifier (bc)[-W]: any classifier of single-label hyperspace
11	MCC	Inference Iterations (ii)[-Iy]: $\{ii \in \mathbb{Z} \mid 1 < ii \leq 100\}$ Chain Iterations (chi)[-Is]: 50% of the probability of chi is 0, and 50% of the probability of chi is a number in $\{chi \in \mathbb{Z} \mid 1 < chi \leq 1500\}$ Payoff function (pof)[-P]): $pof \in$ payoff_function Base classifier (bc)[-W]: any classifier of single-label hyperspace
12	RAkEL	Subsampling value (sv)[-N]: $\{sv \in \mathbb{Z} \mid 0 \leq sv \leq 5\}$ Pruning value (pv)[-P]: $\{pv \in \mathbb{Z} \mid 1 \leq pv \leq 5\}$ Number of labels for each subset (les)[-k]: $\{les \in \mathbb{Z} \mid 1 \leq les \leq \frac{LABELS}{2}\}$ Number of subsets to run in an ensemble (sre)[-M]): $\{sre \in \mathbb{Z} \mid 2 \leq sre \leq min(2 * LABELS, 100)\}$ Base classifier (bc)[-W]: any classifier of single-label hyperspace
13	RAkELd	Subsampling value (sv)[-N]: $\{sv \in \mathbb{Z} \mid 0 \leq sv \leq 5\}$ Pruning value (pv)[-P]: $\{pv \in \mathbb{Z} \mid 1 \leq pv \leq 5\}$

continua na próxima página

#	Algoritmo	Hiperparâmetros
		Batch size (bs)[-batch-size]): $\{bs \in \mathbb{Z} \mid 2 \leq bs \leq min(2 * LABELS, 100)\}$ Base classifier (bc)[-W]: any classifier of single-label hyperspace
14	PCC	Base classifier (bc)[-W]: any classifier of single-label hyperspace
15	PMCC	Inference Iterations (ii)[-Iy]: $\{ii \in \mathbb{Z} \mid 1 < ii \leq 100\}$ Chain Iterations (chi)[-Is]: $\{chi \in \mathbb{Z} \mid 50 \leq chi \leq 1500\}$ Beta (β)[-B]: $\beta \in \mathbb{R} \mid 0.01 \leq 0.99$ step 0.01 Temperature switch (ts)[-O]: $ts \in \{0, 1\}$ Population size (ps)[-M]: $ps \in \mathbb{Z} \mid 1 \leq ps \leq 50$ Payoff function (pof)[-P]): $pof \in$ payoff_function Base classifier (bc)[-W]: any classifier of single-label hyperspace
16	PS	Subsampling value (sv)[-N]: $\{sv \in \mathbb{Z} \mid 0 \leq sv \leq 5\}$ Pruning value (pv)[-P]: $\{pv \in \mathbb{Z} \mid 1 \leq pv \leq 5\}$ Base classifier (bc)[-W]: any classifier of single-label hyperspace
17	PSt	Subsampling value (sv)[-N]: $\{sv \in \mathbb{Z} \mid 0 \leq sv \leq 5\}$ Pruning value (pv)[-P]: $\{pv \in \mathbb{Z} \mid 1 \leq pv \leq 5\}$ Base classifier (bc)[-W]: any classifier of single-label hyperspace
18	RT	Base classifier (bc)[-W]: any classifier of single-label hyperspace

Fonte: Sá et al. (2020)

Tabela 20 – Algoritmos de Classificação Multirrótulo adicionados ao hiperespaço multirrótulo.

#	Algoritmo	Hiperparâmetros
1	MLkNN	Normalize data (norm): $norm \in \{False, True\}$ Number of neighbors (nn): $\{nn \in \mathbb{Z} \mid 1 \leq nn \leq 65\}$
2	HOMER	Normalize data (norm): $norm \in \{False\}$ Method (m): $m \in$ {BalancedClustering, Clustering, Random} Clusters (cl): $\{cl \in \mathbb{Z} \mid 2 \leq cl \leq LABELS + 1$ *if* $LABELS < 9$ *else* $8\}$ Base classifier (bc): $bc \in$ {BinaryRelevance, ClassifierChain, LabelPowerset}
2	ECC	Normalize data (norm): $norm \in \{False\}$

APÊNDICE B

Resultados da Estratégia EMANUEL e dos Algoritmos *Baselines*

Este apêndice apresenta o desempenho de EMANUEL e dos algoritmos *baselines* em relação aos objetivos. As Tabelas 21 e 22 apresentam o desempenho dos algoritmos avaliados em relação ao Macro F-score e ao tamanho do modelo, respectivamente.

Tabela 21 – Resultados do Macro F-score dos algoritmos selecionados da fronteira de Pareto de EMANUEL e dos algoritmos *baselines*. Embora estejamos interessados em ambos os objetivos, destacamos o algoritmo com o melhor Macro F-score para um conjunto de dados. O símbolo "-" indica que não conseguimos encontrar resultados do algoritmo para o conjunto de dados.

Conjunto	**EMANUEL**			**Baseline**					
de Dados	**Min**	**Cent**	**Max**	**BR+AdaBoost**	**CC**	**CDN**	**ECC**	**IBLR+**	**RAkELo**
bibtex	0.010	0.262	0.375	0.280	0.261	0.273	0.326	0.133	0.264
birds	0.000	0.468	0.521	0.233	0.284	0.460	0.373	0.243	0.238
cal500	0.094	0.231	0.235	0.138	0.142	0.140	0.151	0.164	0.112
corel5k	0.001	0.058	0.070	0.033	0.018	0.040	0.042	0.036	0.028
delicious	0.000	0.151	0.168	0.104	0.064	-	0.105	-	0.130
emotions	0.475	0.701	0.710	0.610	0.581	0.680	0.658	0.605	0.560
enron	0.035	0.212	0.262	0.194	0.146	0.231	0.175	0.147	0.178
flags	0.345	0.694	0.713	0.623	0.637	0.632	0.645	0.624	0.605
genbase	0.079	0.782	0.790	0.767	0.763	0.768	0.738	0.736	0.767
mediamill	0.047	0.278	0.378	0.124	0.172	0.243	0.225	0.27	0.180
medical	0.019	0.387	0.442	0.368	0.376	0.268	0.353	0.351	0.365
scene	0.479	0.782	0.812	0.706	0.631	0.778	0.733	0.619	0.637
tmc2007-500	0.119	0.629	0.720	0.529	0.578	0.685	0.625	0.584	0.565
yeast	0.244	0.474	0.486	0.400	0.390	0.411	0.404	0.419	0.345

Tabela 22 – Resultados do tamanho dos modelos induzidos pelos algoritmos selecionados da fronteira de Pareto de EMANUEL e pelos algoritmos *baselines*. Os dados são apresentados em notação científica para melhor visualização. Embora estejamos interessados em ambos os objetivos, destacamos o algoritmo com o melhor tamanho do modelo para um conjunto de dados. O símbolo "-" indica que não conseguimos encontrar resultados do algoritmo para o conjunto de dados.

Conj.	**EMANUEL**			**Baseline**					
Dados	**Min**	**Cent**	**Max**	**BR+AdaBoost**	**CC**	**CDN**	**ECC**	**IBLR+**	**RAkELo**
bibtex	3.79e+05	1.83e+06	3.75e+07	5.24e+06	1.59e+07	4.49e+08	2.37e+08	9.89e+07	5.93e+06
birds	3.02e+04	7.32e+05	1.57e+06	6.31e+05	2.77e+05	5.07e+06	4.04e+06	2.64e+06	2.01e+05
cal500	4.65e+04	3.65e+05	4.10e+06	5.50e+06	2.23e+06	6.19e+07	2.89e+07	1.38e+07	2.95e+06
corel5k	1.46e+05	3.48e+06	7.56e+08	1.13e+07	1.37e+07	5.67e+08	2.09e+08	9.66e+07	8.96e+06
delicious	2.29e+05	4.46e+06	4.88e+07	3.26e+07	6.34e+07	-	9.35e+08	-	3.27e+08
emotions	1.13e+04	8.81e+05	2.59e+07	1.95e+05	8.20e+04	2.98e+06	7.45e+05	4.37e+05	1.65e+05
enron	1.93e+05	1.64e+06	1.65e+07	1.68e+06	2.97e+06	6.62e+07	4.31e+07	1.80e+07	1.69e+06
flags	6.18e+03	1.47e+04	8.86e+04	2.28e+05	3.15e+04	3.05e+06	4.47e+05	1.22e+05	1.24e+05
genbase	2.25e+05	3.15e+05	3.51e+06	1.31e+05	1.67e+06	1.08e+07	2.38e+07	8.13e+06	6.77e+04
mediamill	3.70e+04	2.73e+06	2.95e+07	3.27e+06	7.02e+06	6.95e+08	7.95e+07	5.97e+07	4.02e+07
medical	2.74e+05	3.76e+05	1.46e+07	8.63e+05	3.35e+06	2.82e+07	4.91e+07	1.82e+07	2.36e+05
scene	3.03e+04	1.47e+06	6.07e+07	2.00e+05	1.70e+05	1.01e+07	1.92e+06	4.51e+06	3.12e+05
tmc2007-500	9.86e+04	2.24e+06	1.58e+08	7.30e+05	3.44e+06	5.26e+08	3.78e+07	1.00e+07	1.54e+07
yeast	1.34e+04	2.71e+05	3.11e+07	4.54e+05	3.79e+05	2.19e+07	3.65e+06	2.12e+06	1.42e+06

APÊNDICE C

Experimentos para a Construção dos Modelos Substitutos

Este apêndice descreve os experimentos conduzidos em relação aos modelos substitutos. Esses modelos devem prever os valores de Macro F-score e do logaritmo do tamanho dos modelos, podendo ser do tipo mono-alvo ou multi-alvo e induzidos com os diferentes meta-conjuntos de dados. Para tanto, dois experimentos foram realizados:

1. **Experimento 1**: avaliação o uso dos modelos mono-alvo *versus* multi-alvo; e
2. **Experimento 2**: avaliação do melhor meta-conjunto de dados para a indução dos modelos.

As Seções C.1 e C.2 detalham esses experimentos para os modelos substitutos de EMANUEL$_{SM}$ e de EMANUEL$_{FS+}$, respectivamente.

C.1 Modelos Substitutos de EMANUEL$_{\mathbf{SM}}$

C.1.1 Modelos Mono-alvo x Multi-alvo

O algoritmo de regressão *Random Forest* (RF) foi utilizado para induzir os modelos substitutos de EMANUEL$_{SM}$. O primeiro experimento comparou modelos RF mono-alvo e multi-alvo, com o objetivo de identificar qual versão de modelo apresenta melhor desempenho em termos do coeficiente de determinação (R^2). R^2 fornece um indicativo da qualidade do ajuste do regressor (MILES, 2005; DANGETI, 2017).

Foram realizadas 10 repetições de validação cruzada *10-fold* no conjunto de dados resultante da união dos meta-conjuntos de dados MtDt$_{\mathrm{A}}$ e MtDt$_{\mathrm{R}}$, descritos na Seção

6.2. Para os modelos mono-alvo, calculou-se a média de R^2, utilizando a mesma fórmula para o cálculo de R^2 do modelo multi-alvo. A Tabela 23 apresenta as médias de R^2 dos modelos mono-alvo e multi-alvo. Os valores de R^2 de ambos os modelos são muito próximos. No entanto, os modelos mono-alvo apresentaram R^2 superiores em todos os conjuntos de dados.

Tabela 23 – R^2 médio dos modelos induzidos para EMANUEL$_{SM}$.

#	Conjunto de Dados	Modelo Mono-alvo	Modelo Multi-alvo
1	bibtex	0.986	0.980
2	birds	0.973	0.956
3	cal500	0.968	0.958
4	corel5k	0.979	0.955
5	delicious	0.979	0.973
6	emotions	0.952	0.920
7	enron	0.972	0.958
8	flags	0.943	0.908
9	genbase	0.983	0.966
10	mediamill	0.983	0.977
11	medical	0.983	0.972
12	scene	0.966	0.931
13	tmc2007-500	0.980	0.972
14	yeast	0.951	0.918

O teste não paramétrico de Wilcoxon ($\alpha = 5\%$) foi aplicado para verificar as diferenças estatísticas entre os modelos RF. A hipótese nula assegurava que os modelos mono-alvo e multi-alvo apresentavam R^2 semelhantes. No entanto, a hipótese nula foi rejeitada ($p - value \approx 0.0001$), indicando diferenças estatísticas entre os modelos. Os modelos mono-alvo apresentaram R^2 superiores aos modelos multi-alvo. Assim, dois modelos substitutos foram treinados, um para prever Macro F-score e outro para prever o logaritmo do tamanho do modelo.

C.1.2 Análise dos Meta-conjuntos de Dados

O segundo experimento analisou qual meta-conjunto de dados era mais adequado para treinar os modelos substitutos mono-alvo. Foram utilizados os meta-conjuntos de dados: MtDt$_{\mathrm{A}}$, MtDt$_{\mathrm{R}}$ e MtDt$_{\mathrm{A+R}}$ (Seção 6.2). Para avaliar os modelos substitutos com os diferentes meta-conjuntos de dados, repetiu-se 10 vezes validação cruzada *10-fold*. MtDt$_{\mathrm{A}}$ e MtDt$_{\mathrm{R}}$ foram divididos em k partições. $k-1$ partições foram utilizadas como conjunto de treinamento. Os dados de treinamento para MtDt$_{\mathrm{A+R}}$ foram a união das partições de treinamento de MtDt$_{\mathrm{A}}$ e MtDt$_{\mathrm{R}}$. O conjunto de teste foi o mesmo para todos os meta-conjuntos de dados de treinamento, contendo os dados da k-ésima partição de MtDt$_{\mathrm{A}}$ e MtDt$_{\mathrm{R}}$. As Tabelas 24 e 25 apresentam os valores médios de R^2 dos modelos mono-alvo para Macro F-score e para o logaritmo do tamanho do modelo, respectivamente, quando diferentes meta-conjuntos de dados foram utilizados.

Tabela 24 – R^2 médio dos modelos mono-alvo para Macro F-score quando diferentes meta-conjuntos de dados são utilizados.

#	Conjunto de Dados	$\mathbf{MtDt_R}$	$\mathbf{MtDt_A}$	$\mathbf{MtDt_{A+R}}$
1	bibtex	0.921	0.972	0.981
2	birds	0.881	0.923	0.951
3	cal500	0.817	0.873	0.946
4	corel5k	0.852	0.935	0.965
5	delicious	0.924	0.969	0.981
6	emotions	0.876	0.824	0.909
7	enron	0.917	0.908	0.952
8	flags	0.848	0.817	0.892
9	genbase	0.946	0.938	0.968
10	mediamill	0.944	0.974	0.984
11	medical	0.946	0.940	0.977
12	scene	0.882	0.872	0.939
13	tmc2007-500	0.930	0.949	0.970
14	yeast	0.790	0.797	0.910

Tabela 25 – R^2 médio dos modelos mono-alvo para o logaritmo do tamanho do modelo quando diferentes meta-conjuntos de dados são utilizados.

#	Conjunto de Dados	$\mathbf{MtDt_R}$	$\mathbf{MtDt_A}$	$\mathbf{MtDt_{A+R}}$
1	bibtex	0.968	0.976	0.991
2	birds	0.984	0.988	0.995
3	cal500	0.970	0.950	0.989
4	corel5k	0.982	0.981	0.992
5	delicious	0.933	0.968	0.977
6	emotions	0.981	0.982	0.995
7	enron	0.983	0.979	0.992
8	flags	0.990	0.980	0.994
9	genbase	0.994	0.984	0.997
10	mediamill	0.919	0.966	0.981
11	medical	0.983	0.976	0.989
12	scene	0.987	0.970	0.992
13	tmc2007-500	0.974	0.976	0.989
14	yeast	0.974	0.974	0.992

Analisando as Tabelas 24 e 25, $\mathrm{MtDt_{A+R}}$ apresenta os maiores valores de R^2 para todos os conjuntos de dados, indicando que $\mathrm{MtDt_{A+R}}$ induz modelos mais precisos que os demais meta-conjuntos de dados. Para confirmar essa informação, o teste não paramétrico de Friedman ($\alpha = 5\%$) foi utilizado para verificar a existência de diferença estatística entre os modelos substitutos treinados com os três meta-conjuntos de dados. Dois testes estatísticos foram conduzidos: um para os modelos substitutos que prevêem o logaritmo do tamanho do modelo e outro para os modelos que prevêem o Macro F-score. A hipótese nula de ambos os testes atestava que os modelos substitutos induzidos com diferentes meta-conjuntos de dados apresentavam distribuições similares de R^2.

A hipótese nula foi rejeitada para ambos os testes. Assim, os modelos substitutos apresentaram R^2 diferentes de acordo com o meta-conjunto de dados de treinamento. O teste *post-hoc* de Nemenyi foi utilizado para verificar qual meta-conjunto de dados

gerou o melhor modelo substituto. Os diagramas de diferença crítica das Figuras 77 e 78 apresentam os resultados do teste de Nemenyi. Os modelos mono-alvo que prevêem o Macro F-score e o logaritmo do tamanho do modelo, induzidos com $\mathrm{MtDt_{A+R}}$, foram os que apresentaram os melhores resultados em relação ao R^2. Já os modelos treinados com $\mathrm{MtDt_A}$ e $\mathrm{MtDt_R}$ apresentaram R^2 inferiores, mas equivalentes entre si.

$\mathrm{MtDt_A}$ representa majoritariamente os algoritmos de classificação multirrótulo que participam da evolução do NSGA-II. $\mathrm{MtDt_R}$ tem uma boa distribuição dos algoritmos de classificação multirrótulo do hiperespaço. $\mathrm{MtDt_{A+R}}$ combina as características dos outros dois meta-conjuntos de dados, oferecendo maior representatividade dos algoritmos do hiperespaço. Esse fato pode justificar os resultados obtidos nos testes estatísticos. Logo, $\mathrm{MtDt_{A+R}}$ foi o meta-conjunto de dados utilizado para a indução dos modelos substitutos de $\mathrm{EMANUEL}_{SM}$.

Figura 77 – Diagrama de diferença crítica para os modelos que prevêem Macro F-score, induzidos com diferentes meta-conjuntos de dados.

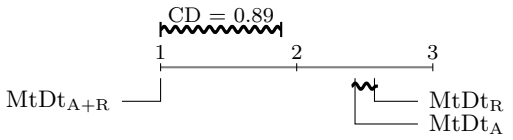
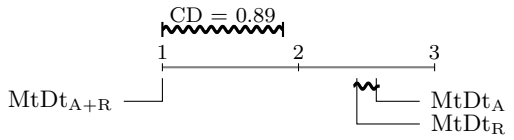


Figura 78 – Diagrama de diferença crítica para modelos que prevêem o logaritmo do tamanho, induzidos com diferentes meta-conjuntos de dados.



C.2 Modelos Substitutos de $\mathrm{EMANUEL_{FS+}}$

C.2.1 Modelos Mono-alvo x Multi-alvo

Os experimentos descritos na Seção C.1 foram replicados para o modelo substituto de $\mathrm{EMANUEL}_{FS+}$, utilizando o algoritmo RF para a indução dos modelos mono-alvo e multi-alvo. A validação cruzada *10-fold* foi repetida 10 vezes com o conjunto de dados resultante da união dos meta-conjuntos de dados $\mathrm{MtDt_A}$ e $\mathrm{MtDt_R}$, descritos na Seção 9.2. Para os modelos mono-alvo, calculou-se o $\mathrm{R^2}$ médio, considerando os valores de $\mathrm{R^2}$ dos modelos que prevêem o Macro F-score e o logaritmo do tamanho do modelo. Para os modelos multi-alvo, o $\mathrm{R^2}$ foi calculado da mesma forma, permitindo a comparação direta entre os modelos treinados. A Tabela 26 apresenta os valores médios de $\mathrm{R^2}$ dos modelos mono-alvo e multi-alvo para os conjuntos de dados avaliados. Na Tabela 26, os modelos mono-alvo possuem os melhores valores para $\mathrm{R^2}$.

O teste não paramétrico de Wilcoxon foi aplicado para avaliar as diferenças estatísticas entre os modelos mono-alvo e multi-alvo ($\alpha = 5\%$). Para esse teste, utilizamos os valores médios de $\mathrm{R^2}$ resultantes das 10 execuções da validação cruzada *10-fold*. A hipótese nula pressupunha a equivalência entre o desempenho ($\mathrm{R^2}$) dos modelos avaliados. Os

Tabela 26 – R^2 médio dos modelos induzidos para EMANUEL$_{FS+}$.

#	Conjunto de Dados	Modelo Mono-alvo	Modelo Multi-alvo
1	birds	0.937	0.929
2	enron	0.902	0.897
3	medical	0.900	0.894
4	scene	0.936	0.930
5	yeast	0.849	0.842

resultados não demonstraram diferença estatística significativa entre os valores de R^2 ($p-value \approx 0.0625$), levando à aceitação da hipótese nula. Apesar de não haver diferenças estatísticas entre os modelos avaliados, os modelos mono-alvo apresentaram desempenho ligeiramente superior aos modelos multi-alvo em termos de R^2, conforme a Tabela 26. Assim, para maximizar a acurácia preditiva dos modelos substitutos de EMANUEL$_{FS+}$, dois modelos mono-alvo foram induzidos: um para prever o Macro F-score e outro para prever o logaritmo do tamanho do modelo.

C.2.2 Análise dos Meta-conjuntos de Dados

O segundo experimento teve por objetivo identificar o meta-conjunto de dados mais adequado para a indução dos modelos substitutos mono-alvo. Três meta-conjuntos de dados distintos foram avaliados: (1) MtDt$_{\mathrm{A}}$, (2) MtDt$_{\mathrm{R}}$ e (3) MtDt$_{\mathrm{A+R}}$ (Seção 9.2). Modelos RF foram induzidos utilizando a abordagem de validação cruzada *10-fold* com 10 repetições. Para MtDt$_{\mathrm{A}}$ e MtDt$_{\mathrm{R}}$, $k-1$ partições foram destinadas ao treinamento. Para MtDt$_{\mathrm{A+R}}$, os dados de treinamento consistiram na união das partições de treino correspondentes de MtDt$_{\mathrm{A}}$ e MtDt$_{\mathrm{R}}$. O conjunto de teste manteve-se constante em todas as configurações, sendo composto pela k-ésima partição tanto de MtDt$_{\mathrm{A}}$ quanto de MtDt$_{\mathrm{R}}$. As Tabelas 27 e 28 apresentam os valores médios de R^2 dos modelos mono-alvo para Macro F-score e para o logaritmo do tamanho do modelo, respectivamente, para os diferentes meta-conjuntos de dados. Em ambas as tabelas, os melhores valores médios de R^2 foram obtidos com MtDt$_{\mathrm{A+R}}$.

Tabela 27 – R^2 médio dos modelos mono-alvo para o Macro F-score quando diferentes meta-conjuntos de dados são utilizados.

#	Conjunto de Dados	MtDt$_{\mathrm{R}}$	MtDt$_{\mathrm{A}}$	MtDt$_{\mathrm{A+R}}$
1	birds	0.892	0.834	0.923
2	enron	0.619	0.754	0.879
3	medical	0.820	0.823	0.888
4	scene	0.863	0.818	0.927
5	yeast	0.740	0.767	0.821

Tabela 28 – R^2 médio dos modelos mono-alvo para o logaritmo do tamanho do modelo quando diferentes meta-conjuntos de dados são utilizados.

#	Conjunto de Dados	$\mathbf{MtDt_R}$	$\mathbf{MtDt_A}$	$\mathbf{MtDt_{A+R}}$
1	birds	0.926	0.902	0.948
2	enron	0.694	0.844	0.922
3	medical	0.859	0.856	0.911
4	scene	0.887	0.828	0.945
5	yeast	0.801	0.824	0.871

Para validar estatisticamente os dados das Tabelas 27 e 28, o teste não paramétrico de Friedman ($\alpha = 5\%$) permitiu avaliar a existência de diferenças significativas no desempenho dos modelos substitutos induzidos com os três meta-conjuntos de dados. Dois testes estatísticos independentes foram conduzidos, um para o logaritmo do tamanho do modelo e outro para o Macro F-score. Para a condução desses testes, foram utilizados os valores médios de R^2 resultantes das 10 execuções da validação cruzada *10-fold* nos três meta-conjuntos de dados. Em ambos os testes, a hipótese nula estabelecia a equivalência nas distribuições de R^2 entre os modelos induzidos a partir dos diferentes meta-conjuntos de dados.

A hipótese nula foi rejeitada para ambos os testes, indicando que os modelos substitutos apresentaram diferenças significativas no R^2 em função do meta-conjunto de dados de treinamento. O teste de Nemenyi foi então utilizado para identificar os grupos de meta-conjuntos de dados com diferenças estatísticas em relação ao R^2. As Figuras 79 e 80 apresentam os diagramas de diferenças críticas para os modelos que prevêem o Macro F-score e o tamanho dos modelos, respectivamente. A análise dessas figuras indica que o $\text{MtDt}_{\text{A+R}}$ pertence ao grupo de meta-conjuntos de dados que induzem os melhores modelos em termos de R^2. Além disso, em ambas as figuras, o $\text{MtDt}_{\text{A+R}}$ possui a melhor classificação no grupo dos melhores meta-conjuntos de dados. Logo, para maximizar a acurácia preditiva dos modelos substitutos de EMANUEL_{FS+}, esses modelos foram induzidos com $\text{MtDt}_{\text{A+R}}$.

Figura 79 – Diagrama de diferença crítica para modelos que prevêem Macro F-score, induzidos com diferentes meta-conjuntos de dados.

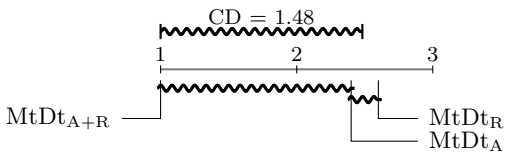
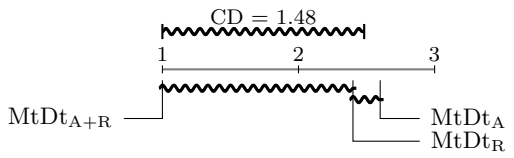


Figura 80 – Diagrama de diferença crítica para modelos que prevêem o logaritmo do tamanho do modelo, induzidos com diferentes meta-conjuntos de dados.



APÊNDICE D

Resultados das estratégias EMANUEL e EMANUEL$_{SM}$ e dos Algoritmos *Baselines*

Este apêndice apresenta resultados relacionados ao tempo de execução das estratégias EMANUEL e EMANUEL$_{SM}$ e ao desempenho dessas estratégias e dos algoritmos *baselines* em relação aos objetivos. A Figura 81 apresenta o tempo de execução das estratégias propostas (sem escala logarítmica) e as Tabelas 29 e 30 o desempenho dos algoritmos avaliados em relação ao Macro F-score e ao tamanho dos modelos induzidos.

Figura 81 – Tempo de execução das estratégias EMANUEL e EMANUEL$_{SM}$ (segundos).

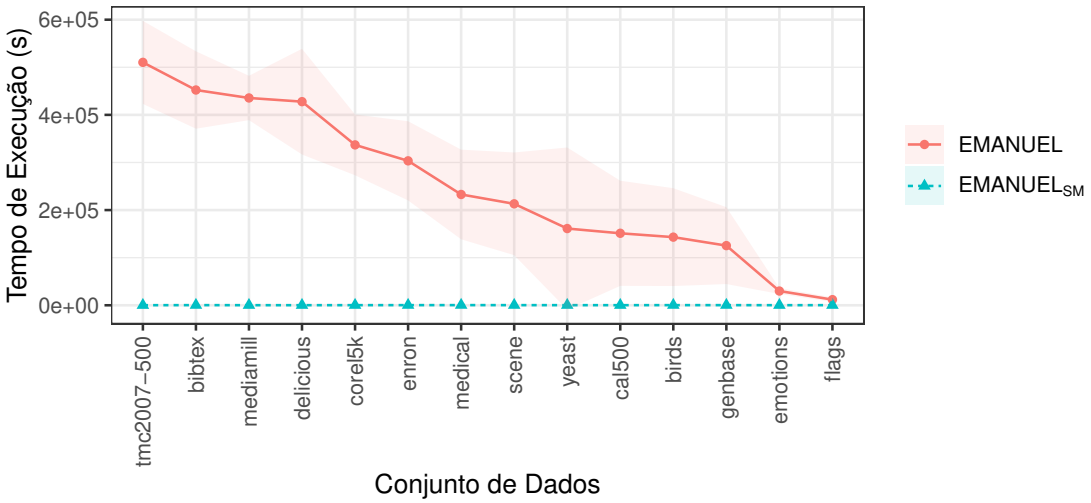


Tabela 29 – Resultados do Macro F-score dos algoritmos selecionados das fronteiras de Pareto de EMANUEL e de EMANUEL$_{SM}$, e dos algoritmos *baselines*. Embora estejamos interessados em ambos os objetivos, identificamos o algoritmo com o melhor Macro F-score para cada conjunto de dados. O símbolo "-" na tabela indica que não foi possível obter resultados para o conjunto de dados.

Conjunto	**EMANUEL**			**EMANUEL$_{SM}$**			**Baseline**							
de Dados	**Min**	**Cent**	**Max**	**Min**	**Cent**	**Max**	**BR+Ada**	**CC**	**CDN**	**ECC**	**IBLR+**	**RAkELo**	**auto-sklearn**	**Auto-MEKA**
bibtex	0.010	0.262	0.375	0.023	0.265	0.388	0.280	0.261	0.273	0.326	0.133	0.264	0.309	0.279
birds	0.000	0.468	0.521	0.000	0.454	0.493	0.233	0.284	0.460	0.373	0.243	0.238	0.26	0.506
cal500	0.094	0.231	0.235	0.094	0.231	0.234	0.138	0.142	0.140	0.151	0.164	0.112	0.183	0.250
corel5k	0.001	0.058	0.070	0.001	0.051	0.061	0.033	0.018	0.040	0.042	0.036	0.028	0.065	0.062
delicious	0.000	0.151	0.168	0.000	0.032	0.151	0.104	0.064	-	0.105	-	0.130	0.150	0.082
emotions	0.475	0.701	0.710	0.519	0.670	0.706	0.610	0.581	0.680	0.658	0.605	0.56	0.628	0.691
enron	0.035	0.212	0.262	0.069	0.138	0.256	0.194	0.146	0.231	0.175	0.147	0.178	0.202	0.259
flags	0.345	0.694	0.713	0.466	0.536	0.694	0.623	0.637	0.632	0.645	0.624	0.605	0.644	0.672
genbase	0.079	0.782	0.790	0.113	0.782	0.789	0.767	0.763	0.768	0.738	0.736	0.767	0.752	0.787
mediamill	0.047	0.278	0.378	0.039	0.278	0.378	0.124	0.172	0.243	0.225	0.270	0.180	0.177	0.359
medical	0.019	0.387	0.442	0.029	0.376	0.402	0.368	0.376	0.268	0.353	0.351	0.365	0.334	0.399
scene	0.479	0.782	0.812	0.442	0.765	0.803	0.706	0.631	0.778	0.733	0.619	0.637	0.747	0.79
tmc2007-500	0.119	0.629	0.720	0.132	0.700	0.720	0.529	0.578	0.685	0.625	0.584	0.565	0.655	0.707
yeast	0.244	0.474	0.486	0.244	0.474	0.486	0.400	0.390	0.411	0.404	0.419	0.345	0.430	0.445

Tabela 30 – Resultados do tamanho dos modelos induzidos pelos algoritmos selecionados das fronteiras de Pareto de EMANUEL e de EMANUEL$_{SM}$, e pelos *baselines*. Os dados são apresentados em notação científica para melhor visualização. Na tabela, BR, ASK e AMK são os *baselines* BR com AdaBoost, `auto-sklearn` e `Auto-MEKA`, respectivamente. Embora estejamos interessados em ambos os objetivos, identificamos o algoritmo com o melhor tamanho de modelo para cada conjunto de dados. O símbolo "-" na tabela indica que não foi possível obter resultados para o conjunto de dados.

Conj.	**EMANUEL**			**EMANUEL$_{SM}$**			**Baseline**							
Dados	**Min**	**Cent**	**Max**	**Min**	**Cent**	**Max**	**BR**	**CC**	**CDN**	**ECC**	**IBLR+**	**RAkELo**	**ASK**	**AMK**
bibtex	3.79e+05	1.83e+06	3.75e+07	3.73e+05	1.41e+06	3.75e+07	5.24e+06	1.59e+07	4.49e+08	2.37e+08	9.89e+07	5.93e+06	1.24e+07	2.81e+07
birds	3.02e+04	7.32e+05	1.57e+06	3.02e+04	9.54e+05	3.04e+07	6.31e+05	2.77e+05	5.07e+06	4.04e+06	2.64e+06	2.01e+05	9.21e+06	7.56e+06
cal500	4.65e+04	3.65e+05	4.10e+06	4.65e+04	3.60e+05	4.10e+06	5.50e+06	2.23e+06	6.19e+07	2.89e+07	1.38e+07	2.95e+06	7.60e+06	7.13e+06
corel5k	1.46e+05	3.48e+06	7.56e+08	1.46e+05	8.46e+05	1.57e+08	1.13e+07	1.37e+07	5.67e+08	2.09e+08	9.66e+07	8.96e+06	7.07e+06	1.97e+08
delicious	2.29e+05	4.46e+06	4.88e+07	2.29e+05	2.72e+05	4.46e+06	3.26e+07	6.34e+07	-	9.35e+08	-	3.27e+08	9.89e+06	6.16e+06
emotions	1.13e+04	8.81e+05	2.59e+07	1.14e+04	3.41e+04	1.04e+07	1.95e+05	8.20e+04	2.98e+06	7.45e+05	4.37e+05	1.65e+05	1.04e+07	9.23e+06
enron	1.93e+05	1.64e+06	1.65e+07	1.92e+05	3.08e+05	5.79e+06	1.68e+06	2.97e+06	6.62e+07	4.31e+07	1.80e+07	1.69e+06	1.84e+07	3.98e+07
flags	6.18e+03	1.47e+04	8.86e+04	6.21e+03	6.79e+03	1.47e+04	2.28e+05	3.15e+04	3.05e+06	4.47e+05	1.22e+05	1.24e+05	5.26e+06	2.73e+06
genbase	2.25e+05	3.15e+05	3.51e+06	2.26e+05	3.15e+05	3.08e+06	1.31e+05	1.67e+06	1.08e+07	2.38e+07	8.13e+06	6.77e+04	1.23e+07	4.80e+07
mediamill	3.70e+04	2.73e+06	2.95e+07	4.07e+04	2.73e+06	2.95e+07	3.27e+06	7.02e+06	6.95e+08	7.95e+07	5.97e+07	4.02e+07	7.86e+09	3.25e+07
medical	2.74e+05	3.76e+05	1.46e+07	2.75e+05	3.75e+05	2.15e+06	8.63e+05	3.35e+06	2.82e+07	4.91e+07	1.82e+07	2.36e+05	1.71e+07	1.21e+07
scene	3.03e+04	1.47e+06	6.07e+07	2.99e+04	1.01e+05	4.79e+06	2.00e+05	1.70e+05	1.01e+07	1.92e+06	4.51e+06	3.12e+05	1.77e+07	2.15e+07
tmc2007-500	9.86e+04	2.24e+06	1.58e+08	9.87e+04	3.73e+07	3.99e+08	7.30e+05	3.44e+06	5.26e+08	3.78e+07	1.00e+07	1.54e+07	5.81e+06	1.50e+08
yeast	1.34e+04	2.71e+05	3.11e+07	1.34e+04	2.71e+05	1.12e+07	4.54e+05	3.79e+05	2.19e+07	3.65e+06	2.12e+06	1.42e+06	1.04e+07	5.89e+08

APÊNDICE E

Algoritmos de Seleção de Atributos do Hiperespaço da Estratégia AutoMLFS

Este apêndice descreve os algoritmos de seleção de atributos multirrótulo do hiperespaço da estratégia AutoMLFS: BRSelectKBest; PPTSelectKBest; BRReliefF; PPTReliefF; MLC_RFE; PPT_RFE; MLCSelectFromModel; e PPTSelectFromModel. Esses algoritmos foram desenvolvidos na linguagem de programação `Python`, sendo adaptações dos algoritmos de seleção de atributos monorrótulo SelectKBest, RFE e SelectFromModel, da biblioteca `scikit-learn` (PEDREGOSA et al., 2011). Além disso, foram implementados algoritmos baseados na função ReliefF da biblioteca `scikit-rebate` (URBANOWICZ et al., 2018). Todos os algoritmos geram um *ranking* dos atributos com base em sua relevância. A seguir, apresentamos uma breve descrição desses algoritmos.

Algoritmos Baseados em SelectKBest

A versão original de SelectkBest (PEDREGOSA et al., 2011) utiliza o teste estatístico qui-quadrado, a informação mútua ou o F-value da ANOVA para calcular os *scores* entre os atributos e o rótulo e, em seguida, seleciona os k atributos com os maiores *scores*. BRSelectKBest e PPTSelectKBest são algoritmos baseados no SelectKBest e utilizam as transformações BR e PPT, respectivamente.

O BRSelectKBest utiliza os mesmos métodos de cálculo de *scores* do SelectKBest. Neste caso, o *score* é calculado entre os atributos e cada rótulo e o *score* final é a média dos *scores* obtidos para cada atributo. Essa média é utilizada para encontrar o *ranking*

dos atributos. Já o PPTSelectKBest transforma o problema multirrótulo em monorrótulo, utilizando a abordagem PPT e, em seguida, utiliza SelectKBest para encontrar o *ranking* dos atributos.

Algoritmos Baseados em ReliefF

ReliefF é uma medida de avaliação da importância dos atributos para o rótulo, comumente utilizada para a seleção de atributos (SPOLAÔR et al., 2013). Desenvolvemos os algoritmos BRReliefF e PPTRelieffF utilizando a função ReliefF. O BRReliefF calcula o *score* entre os atributos e cada um dos rótulos e utiliza a média dos *scores* para o cálculo do *ranking*. Já o PPTRelieffF transforma o problema multirrótulo em monorrótulo, utilizando a transformação PPT. Nesse caso, os *scores* do ReliefF são calculados entre os atributos e os rótulos transformados.

Algoritmos Baseados em RFE

O Algoritmo RFE (PEDREGOSA et al., 2011) utiliza um estimador externo que atribui pesos aos atributos. Inicialmente, o estimador é induzido com todos os atributos. A importância dos atributos é calculada e o atributo menos importante é removido do conjunto de atributos. Este procedimento é repetido de forma recursiva com um conjunto de atributos menor, até que o número de atributos a ser selecionado seja alcançado.

Os algoritmos PPT_RFE e MLC_RFE executam RFE internamente, com os hiperparâmetros:

- ❏ Número de atributos a ser selecionado igual a um, pois isso garante o ranqueamento de todos os atributos; e
- ❏ Um estimador externo que forneça informações sobre a importância dos atributos e possa induzir modelos de classificação tanto monorrótulo quanto multirrótulo, como os algoritmos *Random Forest* e *Decision Trees* da biblioteca `scikit-learn`.

O PPT_RFE emprega a transformação PPT e faz a seleção de atributos monorrótulo utilizando RFE com os hiperparâmetros supracitados. Já o MLC_RFE utiliza diretamente o algoritmo RFE com os mesmos hiperparâmetros. Isso é possível, pois o estimador externo pode lidar com problemas multirrótulo.

Algoritmos Baseados em SelectFromModel

SelectFromModel (PEDREGOSA et al., 2011) também utiliza um estimador externo que atribui uma importância aos atributos e seleciona os atributos mais importantes com base em um limiar. Os algoritmos PPTSelectFromModel e MLCSelectFromModel executam SelectFromModel internamente, tendo como estimador externo um algoritmo que

forneça informações sobre a importância dos atributos e possa induzir modelos de classificação monorrótulo ou multirrótulo. O PPTSelectFromModel utiliza a transformação PPT e seleciona os atributos utilizando o SelectFromModel com um classificador externo monorrótulo. Já o MLCSelectFromModel utiliza diretamente o SelectFromModel com um classificador externo multirrótulo.

APÊNDICE F

Hiperespaço dos Algoritmos de Seleção de Atributos Multirrótulo

O hiperespaço para seleção de atributos multirrótulo contém as opções de não realizar a seleção de atributos (NO_FEATURE_SELECTION) ou de realizar a seleção de atributos utilizando um dos algoritmos do hiperespaço. Caso o AutoML selecione a primeira opção, o algoritmo de classificação multirrótulo é treinado com todos os atributos do conjunto de dados. Caso contrário, a seleção de atributos ocorre com o algoritmo selecionado e o modelo de classificação é induzido com um conjunto de dados com número de atributos reduzido. A probabilidade do AutoML selecionar NO_FEATURE_SELECTION ou cada um dos algoritmos do hiperespaço é igual.

O hiperespaço é composto por 17 algoritmos de seleção de atributos multirrótulo. Nove deles são algoritmos da biblioteca `PyIT-MLFS` (ESKANDARI, 2022), dois algoritmos são da biblioteca `scikit-learn` (PEDREGOSA et al., 2011) e seis dos algoritmos do hiperespaço foram desenvolvidos neste trabalho. Nossas implementações utilizaram as transformações BR e PPT e basearam-se em algoritmos de seleção de atributos monorrótulo da biblioteca `scikit-learn` e na função avaliação de atributos ReliefF da biblioteca `scikit-rebate` (URBANOWICZ et al., 2018). Para mais informações sobre os algoritmos desenvolvidos, consulte a Seção 7.3 e/ou o Apêndice E. A Tabela 31 apresenta os algoritmos de seleção de atributos do hiperespaço, bem como o espaço de hiperparâmetros desses algoritmos.

Tabela 31 – Hiperespaço dos algoritmos de seleção de atributos multirrótulo. Na tabela, FEATURES é o número de atributos do conjunto de dados multirrótulo e o símbolo "-" para biblioteca identifica os algoritmos desenvolvidos neste trabalho.

#	**Algoritmo**	**Biblioteca**	**Hiperparâmetros**
1	D2F	PyIT-MLFS	Number of features (F): $\{F \in \mathbb{Z} \mid \text{F} \in \text{get\_array\_features(FEATURES)}\}$
2	IGMF	PyIT-MLFS	Number of features (F): $\{F \in \mathbb{Z} \mid \text{F} \in \text{get\_array\_features(FEATURES)}\}$
3	LRFS	PyIT-MLFS	Number of features (F): $\{F \in \mathbb{Z} \mid \text{F} \in \text{get\_array\_features(FEATURES)}\}$
4	LSMFS	PyIT-MLFS	Number of features (F): $\{F \in \mathbb{Z} \mid \text{F} \in \text{get\_array\_features(FEATURES)}\}$
5	MDMR	PyIT-MLFS	Number of features (F): $\{F \in \mathbb{Z} \mid \text{F} \in \text{get\_array\_features(FEATURES)}\}$
6	MLSMFS	PyIT-MLFS	Number of features (F): $\{F \in \mathbb{Z} \mid \text{F} \in \text{get\_array\_features(FEATURES)}\}$
7	PMU	PyIT-MLFS	Number of features (F): $\{F \in \mathbb{Z} \mid \text{F} \in \text{get\_array\_features(FEATURES)}\}$
8	PPT-MI	PyIT-MLFS	Number of features (F): $\{F \in \mathbb{Z} \mid \text{F} \in \text{get\_array\_features(FEATURES)}\}$
9	SCLS	PyIT-MLFS	Number of features (F): $\{F \in \mathbb{Z} \mid \text{F} \in \text{get\_array\_features(FEATURES)}\}$
10	SelectFromModel	scikit-learn	Estimator (E): $E \in$ {ExtraTreesClassifier, RandomForestClassifier} Threshold (T): $T \in$ {mean}
11	RFE	scikit-learn	Estimator (E): $E \in$ {ExtraTreesClassifier, RandomForestClassifier} Number of features (F): $\{F \in \mathbb{Z} \mid \text{F} \in \text{get\_array\_features(FEATURES)}\}$
12	BRSelectKBest	-	Number of features (F): $\{F \in \mathbb{Z} \mid \text{F} \in \text{get\_array\_features(FEATURES)}\}$ Method (M): $M \in$ {f_classif, chi2, mutual_info_classif}
13	PPTSelectKBest	-	Number of features (F): $\{F \in \mathbb{Z} \mid \text{F} \in \text{get\_array\_features(FEATURES)}\}$ Method (M): $M \in$ {f_classif, chi2, mutual_info_classif}
14	BRReliefF	-	Number of features (F): $\{F \in \mathbb{Z} \mid \text{F} \in \text{get\_array\_features(FEATURES)}\}$ Neighbors (N): $\{N \in \mathbb{Z} \mid 1 \leq N \leq 10\}$
15	PPTReliefF	-	Number of features (F): $\{F \in \mathbb{Z} \mid \text{F} \in \text{get\_array\_features(FEATURES)}\}$ Neighbors (N): $\{N \in \mathbb{Z} \mid 1 \leq N \leq 10\}$
16	PPT SelectFromModel	-	Estimator (E): $E \in$ {ExtraTreesClassifier, RandomForestClassifier} Number of features (F): $\{F \in \mathbb{Z} \mid \text{F} \in \text{get\_array\_features(FEATURES)}\}$
17	PPT_RFE	-	Estimator (E): $E \in$ {ExtraTreesClassifier, RandomForestClassifier} Number of features (F): $\{F \in \mathbb{Z} \mid \text{F} \in \text{get\_array\_features(FEATURES)}\}$

Assim como no hiperespaço de classificação multirrótulo (Apêndice A), os hiperparâmetros dos algoritmos de seleção de atributos também são representados por meio de vetores. Para espaços de hiperparâmetros numéricos, informamos o intervalo de valores que um hiperparâmetro pode assumir considerando o incremento de 1. Para os algoritmos

que possuem o hiperparâmetro "número de atributos", o vetor de hiperparâmetros é calculado pela função `get_array_features`, definida abaixo. Onde o vetor com os números de atributos a serem selecionados é definido a partir do número de atributos do conjunto de dados (`FEATURES`). Para exemplificar, o conjunto de dados *corel5k* possui 499 atributos e o vetor com números de atributos a serem selecionados tem números entre 149 e 299.

```
function get_array_features(FEATURES)
        inf = sup = 0

        is FEATURES<100
            inf = 0.4
            sup = 0.8
        else if FEATURES>=100 e FEATURES<500
            inf = 0.3
            sup = 0.6
        else if FEATURES>=500 e FEATURES<1000
            inf = 0.2
            sup = 0.4
        else if FEATURES >= 1000
            inf = 0.1
            sup = 0.2

        inf = int(FEATURES*inf)
        sup = int(FEATURES*sup)

        return a vector with integers from inf to sup+1
```

Os algoritmos SelectFromModel, RFE, PPT_SelectFromModel e PPT_RFE do hiperespaço possuem o hiperparâmetro `estimator`, que pode ser um dos algoritmos de classificação da biblioteca `scikit-learn`: ExtraTreesClassifier ou RandomForestClassifier. Para os algoritmos baseados na transformação PPT, o `limiar` é um hiperparâmetro. No entanto, ele não foi incluído no espaço de hiperparâmetros desses algoritmos. Adotamos o `limiar` fixo igual a seis, conforme Eskandari (2022).